Package ‘spatialfusion’

June 12, 2025

Imports methods, graphics, stats, utils, rstan, sp, sf, fields, spam, deldir

Suggests INLA, testthat, tmap, R.rsp, knitr, rmarkdown

Depends R (>= 3.5.0)

Type Package

VignetteBuilder knitr

Additional_repositories `https://inla.r-inla-download.org/R/stable/`

Title Multivariate Analysis of Spatial Data Using a Unifying Spatial Fusion Framework

Version 0.7

Date 2025-05-29

Maintainer Craig Wang <`craigwang247@gmail.com`>

Description Multivariate modelling of geostatistical (point), lattice (areal) and point pattern data in a unifying spatial fusion framework. Details are given in Wang and Furrer (2021) <doi:10.1016/j.csda.2021.107240>. Model inference is done using either 'Stan' <`https://mc-stan.org/`> or 'INLA' <`https://www.r-inla.org/`>.

License GPL (>= 3)

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.2.3

URL `https://www.math.uzh.ch/pages/spatialfusion/`

BugReports `https://git.math.uzh.ch/crwang/spatialfusion/-/issues`

NeedsCompilation no

Author Craig Wang [aut, cre] (ORCID: <`https://orcid.org/0000-0003-1804-2463`>), Reinhard Furrer [ctb] (ORCID: <`https://orcid.org/0000-0002-6319-2332`>), Cincera Annina [ctb]

Repository CRAN

Date/Publication 2025-06-12 13:50:02 UTC

Contents

spatialfusion-package . . . 2
dataDomain . . . 3
dataGeo . . . 4
dataLattice . . . 4
dataPP . . . 5
fitted . . . 5
fusion . . . 6
fusion.dinla . . . 9
fusion.dstan . . . 11
fusionData . . . 14
fusionSimulate . . . 17
plot . . . 19
predict . . . 20
summary . . . 22

Index **24**

`spatialfusion-package` *Multivariate Analysis of Spatial Data Using a Unifying Spatial Fusion Framework*

Description

Multivariate modelling of geostatistical (point), lattice (areal) and point pattern data in a unifying spatial fusion framework. Details are given in Wang and Furrer (2021) <doi:10.1016/j.csda.2021.107240>. Model inference is done using either 'Stan' <https://mc-stan.org/> or 'INLA' <https://www.r-inla.org/>.

Details

Package:	spatialfusion
Type:	Package
Version:	0.7
Date:	2025-05-29
License:	GPL (>= 3)
LazyLoad:	yes

Data analysis pipeline

Preparing data: `fusionData()` is used to set up the data structure needed for spatial fusion modelling. Depending on the chosen `'method'`, either a `dstan` or a `dinla` object is returned. This object is then suppied to the `'data'` argument in `fusion()` for fitting a spatial fusion model. In terms of the `'method'`, Stan provides Hamiltonian Monte Carlo-based full Bayesian inference, while INLA provides approximate Bayesian inferece at a much faster computation speed.

Their results are very similar in our simulation studies (Wang and Furrer, 2021 <doi:10.1016/j.csda.2021.107240>).

IMPORTANT: Users should be familiar with either **rstan** or **INLA** packages themselves. For Stan, users should know how to choose priors appropriately. For INLA, users should know how to set up an appropriate mesh.

Fitting model: `fusion()` is used to fit a spatial fusion model. The most related publication is by Wang and Furrer (2021) <doi:10.1016/j.csda.2021.107240>, which introduced the framework.

We suggest users to test their model on smaller sub-sampled dataset first, to check model fitting issues such as convergence, identifiability etc. It also helps to get an idea of the computation time required. Afterwards, users can fit the model to their full dataset. The output has a class `fusionModel`.

Model diagnostics: Common generic functions such as `fitted()`, `predict()`, `summary()` and `plot()` are available for `fusionModel` objects. Diagnostics of spatial fusion models should be done in the same way as for a Stan or a INLA model, depending on the chosen method.

Author(s)

Craig Wang <craigwang247@gmail.com>

Examples

```
## Citations
citation('spatialfusion')

## Vignette
vignette("spatialfusion_vignette", package = "spatialfusion")
```

`dataDomain` *Municipality map for Canton of Zurich*

Description

This dataset gives the municipality (gemeinde) map for the Canton of Zurich in Switzerland as of 2019, consisting of 162 municipalities.

Usage

```
dataDomain
```

Format

An `sf` data frame containing 162 `Polygons`.

References

https://www.zh.ch/de/politik-staat/statistik-daten.html (Visited: 26/06/2021)

dataGeo *Simulated geostatistical data*

Description

This dataset gives simulated geostatistical data at 200 locations with a normal-distributed response variable and a covariate.

Usage

```
dataGeo
```

Format

An `sf` data frame containing 200 observations with "lungfunction" as the response variable and a "covariate".

dataLattice *Simulated lattice data*

Description

This dataset gives simulated lattice data at 162 areas with a Poisson-distributed response variable, a covariate and an offset term. It has the same set of polygons as `dataDomain`.

Usage

```
dataLattice
```

Format

An `sf` data frame containing 162 observations with "mortality" as the response variable, a "covariate" and a "pop" as the population offset term.

See Also

`dataDomain`

dataPP — *Simulated point pattern data*

Description

This dataset gives the coordinates of 116 events.

Usage

```
dataPP
```

Format

A `SpatialPoints` containing 116 locations with events.

fitted — *Obtain fitted values of spatial fusion model*

Description

Generate fitted values of the response variables based on a spatial fusion model.

Usage

```
## S3 method for class 'fusionModel'
fitted(object, type = c("link", "summary", "full", "latent"), ...)
```

Arguments

`object`	object of class `fusionModel`. Output of `fusion()`.
`type`	string. The default "link" gives the median of linear predictors; "summary" gives the mean, standard deviation and quantiles of linear predictors; "full" gives full marginals for INLA or posterior samples for Stan; "latent" gives the median of latent processes with their corresponding locations.
`...`	additional arguments not used.

Details

For INLA models, no posterior values for point pattern data will be generated.

Value

The returned value is a list containing the fitted results for each response variable.

Author(s)

Craig Wang

See Also

fusion, fusion.dinla, fusion.dstan.

Examples

```
## example based on simulated data

if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 20, n.area = 10, n.grid = 2,
          psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
          point.beta = list(rbind(1,5)),
          area.beta = list(rbind(-1, 0.5)),
          distributions = c("normal","poisson"),
          design.mat = matrix(c(1,1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <-cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                    cov.area = dat$data$X_area[,2]))

dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
                pp.data = dat$data$lgcp.coords[[1]],
                distributions = c("normal","poisson"),
                method = "INLA")

mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
                prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
                mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))

fit_inla <- fitted(mod_inla, type = "summary")
}
```

`fusion` *Fit a spatial fusion model*

Description

Fit a spatial fusion model based on the unifying framework proposed by Wang and Furrer (2021). One or more latent Gaussian process(es) is assumed to be associated with the spatial response variables.

Usage

```
fusion(data, n.latent = 1, bans = 0, pp.offset,
       verbose = FALSE, ...)
```

Arguments

`data`	an object of class either `dstan` or `dinla`. Output of `fusionData()`.
`n.latent`	integer. Number of latent processes to be modeled.
`bans`	either 0 or a matrix of 0s and 1s with dimension J times n.latent, where J is the total number of response variables. If `matrix`, 1 indicates banning an association between the latent process and response variable. If 0, no association is banned.
`pp.offset`	numeric, vector of numeric or matrix of numeric. Offset term for point pattern data.
`verbose`	logical. If TRUE, prints progress and debugging information.
`...`	additional arguments depending on the class of `data`

Details

It is not possible to add covariates for point pattern data. However, an offset term can be supplied. Any covariate information can be taken into account by firstly fit a fixed effect model and enter the fitted values into the offset term as `pp.offset`.

Value

The returned value is a named list of class `fusionModel` consisting of model output and data structure used. If the model is fitted with INLA, the mesh used is also included.

Author(s)

Craig Wang

References

Wang, C., Furrer, R. and for the SNC Study Group (2021). Combining heterogeneous spatial datasets with process-based spatial fusion models: A unifying framework. *Computational Statistics & Data Analysis*, 161, 107240.

See Also

`fusion.dinla`, `fusion.dstan`, `fusionData` for preparing data, `fitted.fusionModel` for extracting fitted values, `predict.fusionModel` for prediction.

Examples

```
## example based on simulated data

dat <- fusionSimulate(n.point = 20, n.area = 10,
          psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
```

```
          point.beta = list(rbind(1,5)),
          area.beta = list(rbind(-1, 0.5)),
          distributions = c("normal","poisson"),
          design.mat = matrix(c(1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                     cov.area = dat$data$X_area[,2]))

dat_stan <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
                distributions = c("normal","poisson"),
                method = "Stan")

## S3 method for class 'dstan'
mod_stan <- fusion(data = dat_stan, n.latent = 1, bans = 0,
                prior.phi = list(distr = "normal", pars = c(1, 10)),
                nsamples = 1000, nburnin = 500, nchain = 1, ncore = 1)
summary(mod_stan)



if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 20, n.area = 10, n.grid = 10, n.pred =15,
          psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
          point.beta = list(rbind(1,5)),
          area.beta = list(rbind(-1, 0.5)), nvar.pp = 1,
          distributions = c("normal","poisson"),
          design.mat = matrix(c(1,1,1), ncol=1), pp.offset = 1)

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                     cov.area = dat$data$X_area[,2]))

dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
              pp.data = dat$data$lgcp.coords[[1]], distributions = c("normal","poisson"),
                method = "INLA")


## S3 method for class 'dinla'
mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
                prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
```

```
                mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))
summary(mod_inla)
}
```

`fusion.dinla` *Fit a spatial fusion model using INLA*

Description

Fit a spatial fusion model using INLA based on the unifying framework proposed by Wang and Furrer (2021). One or more latent Gaussian process(es) is assumed to be associated with the spatial response variables.

Usage

```
## S3 method for class 'dinla'
fusion(data, n.latent = 1, bans = 0, pp.offset,
    verbose = FALSE, alpha = 3/2, prior.range,
    prior.sigma, prior.args, mesh.locs, mesh.max.edge,
    mesh.args, inla.args, ...)
```

Arguments

`data`	an object of class `dinla`. Output of `fusionData()`.
`n.latent`	integer. Number of latent processes to be modeled.
`bans`	either 0 or a matrix of 0s and 1s with dimension J times n.latent, where J is the total number of response variables. If `matrix`, 1 indicates banning an association between the latent process and response variable. If 0, no association is banned.
`pp.offset`	numeric, vector of numeric or matrix of numeric. Offset term for point pattern data.
`verbose`	logical. If TRUE, prints progress and debugging information
`alpha`	numeric between 0 and 2. Determines the covariance model, defined as $\nu + 1$ for two dimensional space. Default value is 3/2 which corresponds to the exponential covariance model. See details.
`prior.range`	vector of length 2, with (range0, Prange) specifying that P($\rho\sqrt{8\nu}$ < range0) = Prange, where $\rho\sqrt{8\nu}$ is the practical spatial range of the random field. If Prange is NA, then range0 is used as a fixed range value. See details.
`prior.sigma`	vector of length 2, with (sigma0, Psigma) specifying that P(σ > sigma0) = Psigma, where σ is the marginal standard deviation of the field. If Psigma is NA, then sigma0 is used as a fixed sigma value. See details.
`prior.args`	named list. Other prior arguments for `inla.spde2.matern()` in **INLA**.
`mesh.locs`	matrix with two columns, or a `SpatialPoints`, `SpatialPointsDataFrame` object. Locations to be used as initial triangulation nodes.

`mesh.max.edge`	vector of length one or two. The largest allowed triangle edge length for inner (and optional outer extension) mesh.
`mesh.args`	named list. Other mesh arguments passed to `inla.mesh.2d()` in **INLA**.
`inla.args`	named list. Other inla arguments passed to `inla()` **INLA**.
`...`	additional arguments not used

Details

The prior used for modeling the latent spatial processes is `inla.spde2.matern`. Each spatial component is named as `sij`, where `i` denotes the ith latent process and `j` denotes the jth variable. For example, `s12` is the first latent process that is associated with the second variable. The first variable (with the following ordering: geostatistical, lattice, point pattern data) that a spatial component is associated with will have the original component, then the subsequent spatial components associated with other variables are treated as 'copies' of the original component modified by a coefficient `Beta`, as one of the latent parameters.

The INLA approximation only works for Matern covariance function, which can be written as

$C(d) = \sigma^2/(2^{\nu-1}\Gamma(\nu)) * (d\sqrt{2\nu}/\rho)^{\nu}K_{\nu}(d\sqrt{2\nu}/\rho),$

where d is the Euclidean distance, K_ν is a modified Bessel function, ρ is the spatial range, σ^2 is the partial sill and ν is the smoothness parameter. NOTE: the range parameter in INLA output is defined as "practical range" as $\rho\sqrt{8\nu}$.

Value

The returned value is a list consists of

`model`	an object of class `inla` representing the fitted INLA model
`mesh`	an object of class `inla.mesh` containing the mesh used.
`data`	the data structure used to fit the model

Author(s)

Craig Wang

References

Wang, C., Furrer, R. and for the SNC Study Group (2021). Combining heterogeneous spatial datasets with process-based spatial fusion models: A unifying framework. *Computational Statistics & Data Analysis*, 161, 107240.

See Also

`fusionData` for preparing data, `fitted` for extracting fitted values, `predict` for prediction.

Examples

```
## example based on simulated data

if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 50, n.area = 20, n.grid = 4,
               psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
               point.beta = list(rbind(1,5)),
               area.beta = list(rbind(-1, 0.5)),
               distributions = c("normal","poisson"),
               design.mat = matrix(c(1,1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                     cov.area = dat$data$X_area[,2]))

dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
              pp.data = dat$data$lgcp.coords[[1]], distributions = c("normal","poisson"),
                method = "INLA")

mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
                prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
                mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))

summary(mod_inla)
}
```

`fusion.dstan` *Fit a spatial fusion model using Stan*

Description

Fit a spatial fusion model using Stan based on the unifying framework proposed by Wang and Furrer (2021). One or more latent Gaussian process(es) is assumed to be associated with the spatial response variables.

Usage

```
## S3 method for class 'dstan'
fusion(data, n.latent = 1, bans = 0, pp.offset,
    verbose = FALSE, prior.pointbeta, prior.areabeta,
    prior.tausq, prior.phi, prior.z,
    nsamples = 2000, nburnin = 1000, thinning = 1,
    nchain = 2, ncore = 2, adapt.delta = 0.95, ...)
```

Arguments

- `data` — an object of class `dstan`. Output of `fusionData()`.
- `n.latent` — integer. Number of latent processes to be modeled.
- `bans` — either 0 or a matrix of 0s and 1s with dimension J times n.latent, where J is the total number of response variables. If `matrix`, 1 indicates banning an association between the latent process and response variable.If 0, no association is banned.
- `pp.offset` — numeric, vector of numeric or matrix of numeric. Offset term for point pattern data.
- `verbose` — logical. If TRUE, prints progress and debugging information.
- `prior.pointbeta` — a list with prior information for the coeffients of geostatistical model component. The default prior is `list(distr = "normal", pars = c(0, 10))`, i.e. a normal distribution with mean 0 and standard devation 10.
- `prior.areabeta` — a list with prior information for the coeffients of lattice model component. The default prior is `list(distr = "normal", pars = c(0, 10))`, i.e. a normal distribution with mean 0 and standard devation 10.
- `prior.tausq` — a list with prior information for the coeffients of geostatistical model component. The default prior is `list(distr = "inv_gamma", pars = c(2, 1))`, i.e. a inverse gamma distribution with shape 2 and rate 1.
- `prior.phi` — a list with prior information for the spatial range parameter. NO default prior is available. We recomend using a moderately informative normal prior.
- `prior.z` — a list with prior information for the design matrix, which also controls the partial sill. The default prior is `list(distr = "normal", pars = c(1, 1))`, i.e. a normal distribution with mean 1 and standard devation 1.
- `nsamples` — a positive integer specifying the number of samples for each chain (including burn-in samples). Default 2000.
- `nburnin` — a positive integer specifying the number of burn-in samples. Default 1000.
- `thinning` — a positive integer specifying the thinning parameter. Default 1.
- `nchain` — a positive integer specifying the number of chains. Default 2.
- `ncore` — a positive integer specifying the number of cores to use when executing the chains in parallel. Default 2.
- `adapt.delta` — a numeric value between 0 and 1 specifying the target acceptance rate. Default 0.95.
- `...` — additional arguments passed to `sampling` function in **rstan**

Details

In the model parameterization, `beta` are fixed-effect coefficients, `phi` is the range parameter, `Z_ij` is the ith row and j column of the design matrix for latent processes and `tau_sq` is the variance parameter of a normal distribution.

Value

The returned value is a list consists of

`model`	an object of S4 class `stanfit` representing the fitted Stan model
`data`	the data structure used to fit the model

Note

Only exponential covariance model for the latent processes is implemented. However, it can be easily extended by modifying the model code from the output.

Author(s)

Craig Wang

References

Wang, C., Furrer, R. and for the SNC Study Group (2021). Combining heterogeneous spatial datasets with process-based spatial fusion models: A unifying framework. *Computational Statistics & Data Analysis*, 161, 107240.

See Also

`fusion.dinla`, `fusion.dstan`, `fusionData` for preparing data, `fitted.fusionModel` for extracting fitted values, `predict.fusionModel` for prediction.

Examples

```
## example based on simulated data

dat <- fusionSimulate(n.point = 20, n.area = 10,
          psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
          point.beta = list(rbind(1,5)),
          area.beta = list(rbind(-1, 0.5)),
          distributions = c("normal","poisson"),
          design.mat = matrix(c(1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                    data.frame(outcome = dat$data$Y_area[[1]],
                    cov.area = dat$data$X_area[,2]))

dat_stan <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
                distributions = c("normal","poisson"),
                method = "Stan")
```

```
## S3 method for class 'dstan'
mod_stan <- fusion(data = dat_stan, n.latent = 1, bans = 0,
                   prior.phi = list(distr = "normal", pars = c(1, 10)),
                   nsamples = 1000, nburnin = 500, nchain = 1, ncore = 1)
summary(mod_stan)
```

---

fusionData *Prepare data structure for spatial fusion modelling*

Description

Takes various datasets and formulas from different spatial data types and process them to prepare for spatial fusion modeling using either Stan or INLA.

Usage

```
fusionData(geo.data, geo.formula,
           lattice.data, lattice.formula,
           pp.data, distributions, domain = NULL,
           method = c("Stan", "INLA"),
           proj4string = CRS(as.character(NA)),
           stan.control = NULL)
```

Arguments

`geo.data`	an object of class `data.frame` or `sf`. If `data.frame`, it must have column names "x" and "y" as coordinates of observations.
`geo.formula`	an object of class `formula`. A symbolic description of the model to be fitted for geostatistical data. For multivariate geostatistical data, use syntax `cbind(y1, y2)` followed by ~.
`lattice.data`	an object of class `sf`. Contains lattice data.
`lattice.formula`	an object of class `formula`. A symbolic description of the model to be fitted for lattice data. For multivariate lattice data, use syntax `cbind(y1, y2)` followed by ~.
`pp.data`	an object of class `data.frame` or `sf`, or a list of them. If `data.frame`, it must have column names "x" and "y" as coordinates.
`distributions`	a vector of strings. Specifying the distributions of each geostatistical and lattice response variable, currently "Gaussian" or "normal", "Poisson" (count) and "Bernoulli" (binary) are supported. Note: no distribution is required to be specified for point pattern data.
`domain`	an object of class `sf`. The spatial domain considered for computing gridded point pattern data. If `NULL`, a bounding box that contains all spatial units is used.
`method`	character. Either 'Stan' or 'INLA', the method to be used for fitting the spatial fusion model later.

`proj4string` projection string of class `CRS-class`.

`stan.control` a named list of parameters to control the Stan implementation of spatial fusion models. Default to NULL such that all the default values are used.

- `n.neighbor` (positive integer) Number of nearest neighbors to consider. Default to 5.
- `n.sampling` (positive integer) Number of sampling points for each area. Default to 5.
- `n.grid` (positive integer) Number of grid used to divide the spatial domain in each of x- and y-direction to count the number of cases/events in each grid. Default to 10.

Details

It is not possible to add covariate for point pattern data in the spatial fusion framework. However, an offset term can be supplied to `pp.offset` in the modelling stage with `fusion`. Any covariate information can be taken into account by firstly fit a fixed effect model and enter the fitted values into the offset term.

Value

The returned value is an object of either class `dstan` or `dinla`, depending on the chosen `method`. They are both lists that contain:

`distributions` distribution specified each response variable.

`n_point` sample size for geostatistical data.

`n_area` sample size for lattice data.

`n_grid` Set to 1 for INLA, set to the number of grids for Stan.

`p_point` number of coefficients for geostatistical model component (only if there is geostatistical data).

`n_point_var`, `n_area_var`, `n_pp_var`
number of response variables for each data type.

`Y_point` response variable for geostatistical data (only if there is geostatistical data).

`X_point` covariates for geostatistical data (only if there is geostatistical data).

`p_area` number of coefficients for lattice model component (only if there is lattice data).

`Y_area` response variable for lattice data (only if there is lattice data).

`X_area` covariates for lattice data (only if there is lattice data).

`geo.formula`, `lattice.formula`
formulas used for geostatistical and lattice data.

`dstan` additionally contains:

`n_neighbor` number of nearest neighbors to consider for NNGP modelling.

`n_sample` total number of sampling points.

`nearid`, `nearind_sample`
vectors containing neighborhood indices

C_nei, C_site_nei, sC_nei, sC_site_nei
: various distance matrices

A1
: aggregation matrix that maps sampling points to areal averages (only if there is lattice data).

Y_pp
: the number of cases/events in each grid for point pattern data (only if there is point pattern data).

area
: the area of each grid (only if there is point pattern data).

grd_lrg
: the grid generated for point pattern data modeling (only if there is point pattern data).

locs
: all the locations where the latent components are modelled.

dinla additionally contains:

domain
: spatial domain as a SpatialPolygons-class

locs_point
: locations of geostatistical data.

locs_pp
: locations of point pattern data.

poly
: lattice data as a SpatialPolygonsDataFrame-class.

Author(s)

Craig Wang

See Also

fusion.dinla, fusion.dstan

Examples

```
## example based on simulated built-in data

dat <- fusionData(dataGeo, lungfunction ~ covariate,
           dataLattice, mortality ~ covariate,
           dataPP, distribution = c("normal","poisson"),
           domain = dataDomain,
           method = "INLA")


if (requireNamespace("INLA", quietly = TRUE)) {
## fit a spatial fusion model on the prepared data
## pp.offset = 400 was chosen based on simulation parameters
mod <- fusion(data = dat, n.latent = 1, bans = 0, pp.offset = 400,
           prior.range = c(0.1, 0.5), prior.sigma = c(1, 0.5),
           mesh.locs = dat$locs_point, mesh.max.edge = c(0.5, 1))

## parameter estimates
summary(mod)
}
```

fusionSimulate *Simulate spatial data*

Description

Simulate spatial response variables with different data types, including geostatistical (point), lattice (areal), and point pattern data. They share common latent Gaussian processes. The geostatistical and lattice response variables are allowed to have fixed effects.

Usage

```
fusionSimulate(n.point, n.area, n.grid, n.pred, dimension = 10,
               psill = 5, phi = 1, nugget = 0, tau.sq = 1,
               domain = NULL, point.beta = NULL, area.beta = NULL,
               nvar.pp = 1, distributions,
               design.mat = matrix(c(1, 1.5, 2)),
               pp.offset, seed)
```

Arguments

`n.point`	positive integer. Sample size for geostatistical (point) data.
`n.area`	positive integer. Sample size for lattice (areal) data.
`n.grid`	positive integer. Number of grid to be divided in each direction of the spatial domain.
`n.pred`	positive integer. Number of prediction locations to sample regularly.
`dimension`	positive integer. Dimension of the square spatial domain.
`domain`	an object of class `sf`. The spatial domain considered for the simulation. If NULL, a square domain with length `dimension` is used.
`psill`	positive numeric or numeric vector. Partial sill parameter(s) of the latent Gaussian process(es).
`phi`	positive numeric or numeric vector. Range parameter(s) of the latent Gaussian process(es).
`nugget`	positive numeric vector. Nugget parameter(s) of the latent Gaussian process(es).
`tau.sq`	positive numeric vector. Variance component(s) for normally distributed responses.
`point.beta`	a list of column matrices. Regression coefficient for geostatistical data, e.g. `list(rbind(1,2,3),rbind(2,4,6))` for two geostatistical response variables with an intercept plus two covariates each.
`area.beta`	a list of column matrices. Regression coefficient for lattice data, e.g. see above.
`nvar.pp`	numeric. Number of point pattern response variables to simulate.
`distributions`	character vector. Names of distributions for each dependent variables with geostatistical and lattice type, currently “Gaussian” or “normal”, “Poisson” (count) and “Bernoulli” (binary) are supported. Note: no distribution for point pattern data is required to be specified.

`design.mat` matrix. Design matrix for the latent Gaussian process(es), with the number of columns equal to the number of latent processes.

`pp.offset` numeric. A single offset term for the intensity of point pattern data.

`seed` integer. Random seed.

Details

The exponential covariance model is used,

$C(d) = \sigma^2 \exp -d/\phi$

where d is the Euclidean distance, σ^2 is the partial sill and ϕ is the spatial range.

If the purpose is to validate a fitted latent spatial components of a spatial fusion model, one can check the fitted latent values against `mrf[sample.ind, -1:2]`. If the purpose is to investigate prediction performance of latent spatial components, one can predict at locations `pred.loc` and check against `mrf[pred.ind, -1:2]`.

Value

The returned value is a list that consists of:

`data` a named list providing data variables.

`mrf` a data.frame of locations and the latent Gaussian process.

`domain` an object of class `sf` describing the whole domain.

`pred.loc` a data.frame of locations for prediction.

`pred.ind` a vector of indices for prediction locations.

`sample.ind` a vector providing the indices of sampled locations in the Gaussian process. (only if there is geostatistical data)

`mean.w` a list of aggregated latent process for each area. (only if there is lattice data)

`poly` an object of class `sf` for the lattice data. (only if there is lattice data)

`lgcp.grid` a data.frame containing the centroids of gridded cells for point pattern data and the corresponding event counts. (only if there is point pattern data)

Author(s)

Craig Wang

See Also

`fusion`, `fusion.dstan`

Examples

```
# three responses with a single latent Gaussian process
dat1 <- fusionSimulate(n.point = 50, n.area = 5, n.grid = 2,
            psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
            point.beta = list(rbind(1,5)), area.beta = list(rbind(-1, 0.5)),
            distributions = c("normal","poisson"), pp.offset = 1,
```

```
              design.mat = matrix(c(1,1,1)))


# three responses with two latent Gaussian processes
dat2 <- fusionSimulate(n.point = 50, n.area = 5, n.grid = 2,
               psill = c(1,2), phi = c(2,1), nugget = c(0,0), tau.sq = 1,
               point.beta = list(rbind(1,5)), area.beta = list(rbind(-1, 0.5)),
               distributions = c("normal","poisson"), pp.offset = 0.1,
               design.mat = matrix(c(1,1,1,2,3,4), ncol = 2))
```

`plot` *Generate diagnostics plot for a fusion model*

Description

Plot model diagnostics for `fusionModel` objects. By default, it shows posterior versus prior distributions of fixed effect coefficients and latent parameters. The names of fixed effect coefficients are covariate names followed by internal parameter names in parentheses. 'beta_p' denotes the coefficients for point data and 'beta_a' denotes the coefficients for lattice data.

Usage

```
## S3 method for class 'fusionModel'
plot(x, posterior = TRUE, interactive = TRUE, ...)
```

Arguments

`x`	object of class `fusionModel`. Output of `fusion()`.
`posterior`	logical. If `TRUE`, then shows posterior versus prior distributions of fixed effect coefficients and latent parameters.
`interactive`	logical. If `TRUE`, then print messages in the terminal to proceed to next plots.
`...`	additional arguments not used

Details

When `posterior = FALSE`, then traceplot of posterior samples for the fixed effect coefficients and latent parameters are shown for Stan approach and the mesh overlayed with spatial data is shown for INLA approach.

Value

No return value, called for side effects

Author(s)

Craig Wang

Examples

```
## example based on simulated data

if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 20, n.area = 10, n.grid = 2,
               psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
               point.beta = list(rbind(1,5)),
               area.beta = list(rbind(-1, 0.5)),
               distributions = c("normal","poisson"),
               design.mat = matrix(c(1,1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                       y = dat$mrf[dat$sample.ind, "y"],
                       cov.point = dat$data$X_point[,2],
                       outcome = dat$data$Y_point[[1]])
lattice_data <- cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                    cov.area = dat$data$X_area[,2]))


dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                   lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
              pp.data = dat$data$lgcp.coords[[1]], distributions = c("normal","poisson"),
                   method = "INLA")

mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
                   prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
                   mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))

plot(mod_inla, interactive = FALSE)
}
```

`predict` *Obtain predictions for the latent processes of spatial fusion model*

Description

Generate posterior values containing predictions of the latent Gaussian process(es) based on a fitted spatial fusion model and new locations.

Usage

```
## S3 method for class 'fusionModel'
predict(object, new.locs, type = c("summary", "full"), ...)
```

Arguments

- `object` — an object of class `fusionModel`. Output of `fusion()`.
- `new.locs` — of class `data.frame` or `sf`. Contains the locations where the latent process(es) will be predicted. If `data.frame`, it must have column names "x" and "y" as coordinates of observations.
- `type` — string, The default "summary" gives posterior median of latent process(es); "full" gives full marginals (for INLA) or posterior samples (for Stan) of latent process(es).
- `...` — additional arguments not used

Value

The returned value is an object of class `sf` containing the posterior values for the latent spatial components and the prediction location in the geometry column of type `POINTS`.

For INLA models, the output represents the latent components that are associated with each response variable multiplied by the design matrix Z. They are indexed with `ij`, where `i` denotes the ith latent process and `j` denotes the jth variable. The variables are ordered by geostatistical, lattice, point pattern data.

For Stan models, the output represents the original latent components before multiplied by the deisng matrix Z. Each spatial component is indexed with `i`, where `i` denotes the ith latent process.

Author(s)

Craig Wang

Examples

```
## example based on simulated data

if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 20, n.area = 10, n.grid = 2,
          psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
          point.beta = list(rbind(1,5)),
          area.beta = list(rbind(-1, 0.5)),
          distributions = c("normal","poisson"),
          design.mat = matrix(c(1,1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                y = dat$mrf[dat$sample.ind, "y"],
                cov.point = dat$data$X_point[,2],
                outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                     data.frame(outcome = dat$data$Y_area[[1]],
                     cov.area = dat$data$X_area[,2]))

dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
                pp.data = dat$data$lgcp.coords[[1]],
```

```
                distributions = c("normal","poisson"), method = "INLA")

mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
            prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
            mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))

pred_inla <- predict(mod_inla, dat$pred.loc, type = "summary")
}
```

`summary` *Obtain summary of parameter estimates for a spatial fusion model*

Description

Generate summary statistics for posterior parameter estimates from a spatial fusion model.

Usage

```
## S3 method for class 'fusionModel'
summary(object, digits = 3, ...)
```

Arguments

`object`	object of class `fusionModel`. Output of `fusion()`.
`digits`	integer. The number of significant digits.
`...`	additional arguments not used.

Value

The returned value is a matrix containing the parameter estimates and their summary statistics. The names of fixed effect coefficients are covariate names followed by internal parameter names in parentheses. 'beta_p' denotes the coefficients for point data and 'beta_a' denotes the coefficients for lattice data.

Author(s)

Craig Wang

Examples

```
## example based on simulated data

if (requireNamespace("INLA", quietly = TRUE)) {
dat <- fusionSimulate(n.point = 20, n.area = 10, n.grid = 2,
       psill = 1, phi = 1, nugget = 0, tau.sq = 0.5,
       point.beta = list(rbind(1,5)),
       area.beta = list(rbind(-1, 0.5)),
```

```
        distributions = c("normal","poisson"),
        design.mat = matrix(c(1,1,1)))

geo_data <- data.frame(x = dat$mrf[dat$sample.ind, "x"],
                       y = dat$mrf[dat$sample.ind, "y"],
                       cov.point = dat$data$X_point[,2],
                       outcome = dat$data$Y_point[[1]])

lattice_data <- cbind(dat$poly,
                      data.frame(outcome = dat$data$Y_area[[1]],
                     cov.area = dat$data$X_area[,2]))

dat_inla <- fusionData(geo.data = geo_data, geo.formula = outcome ~ cov.point,
                lattice.data = lattice_data, lattice.formula = outcome ~ cov.area,
                pp.data = dat$data$lgcp.coords[[1]],
                distributions = c("normal","poisson"), method = "INLA")

mod_inla <- fusion(data = dat_inla, n.latent = 1, bans = 0,
                prior.range = c(1, 0.5), prior.sigma = c(1, 0.5),
                mesh.locs = dat_inla$locs_point, mesh.max.edge = c(0.5, 1))

summary(mod_inla)
}
```

Index

∗ **aplot**
plot, 19
∗ **datagen**
fusionSimulate, 17
∗ **datasets**
dataDomain, 3
dataGeo, 4
dataLattice, 4
dataPP, 5
∗ **methods**
fitted, 5
fusionData, 14
predict, 20
summary, 22
∗ **models**
fusion, 6
fusion.dinla, 9
fusion.dstan, 11
∗ **package**
spatialfusion-package, 2

data.frame, *14*
dataDomain, 3, *4*
dataGeo, 4
dataLattice, 4
dataPP, 5

fitted, 5, *10*
fitted.fusionModel, *7*, *13*
formula, *14*
fusion, *5, 6*, 6, *15*, *18, 19*, *21, 22*
fusion.dinla, *6, 7*, 9, *13*, *16*
fusion.dstan, *6, 7*, 11, *13*, *16*, *18*
fusionData, *7*, *9, 10*, *12, 13*, 14
fusionSimulate, 17

plot, 19
predict, *10*, 20
predict.fusionModel, *7*, *13*
print.dinla *(*fusionData*)*, 14
print.dstan *(*fusionData*)*, 14

spatialfusion *(*spatialfusion-package*)*, 2
spatialfusion-package, 2
SpatialPolygons-class, *16*
SpatialPolygonsDataFrame-class, *16*
summary, 22