PDF (Please read the PDF to see the equations.)

SpaceTimeModels - An interface for fitting parametric spatial and spatio-temporal models with R and INLA

Introduction

The `SpaceTimeModels` R package provides a simplified interface for parametrizing a few standard spatial and spatio-temporal models with R-INLA. The models enable parametric (ie. the parameters are estimated from the data) smoothing over space and time and determining effect of covariates on response.

Autocorrelation

Things close in space and time often resemble each other. We can observe this e.g. in nature where large scale processes such as climate and geomorphic processes shape biological processes. For example, coniferous trees have been adapted to cool climate and found more than deciduous that are more prevalent in warm climate conditions. Due to dependencies in the underlying processes, measurements obtained close to each other tend to predict each other better than distant ones. This degree of similarity is known as autocorrelation, i.e. the correlation within a process itself.

From the statistical modelling point of view, dependency in data can cause estimates obtained from a model with assumed indepedence to be biased, e.g. the model provides too small p-values. Several approaches for spatial and spatio-temporal data have been developed to take autocorrelation properly into account.

Since there often are no repeated measurements available that would allow estimating autocorrelation structure directly, several assumptions are made. These often include specifying a neighborhood structure (e.g. the observations depend on each other as a function of distance), stationarity (variance is constant across space/time) and isotropy (independence of direction in space).

Discrete and continuous data

Data in space and time can be indexed in several ways. Discrete data is defined in a subset of specific values such as time in years indexed $t = 1, 2, \ldots, T$ or areas $s = 1, 2, \ldots, S$ separated by boundaries. Continous data is index by any values within a range such as $t \in [1, T]$, where there are infinitely many time points within this range. Similarly, continuous space has infinitely many locations enclosed by some boundary (like a study area) Ω, i.e. $s \in \Omega$. The package provides model classes for combinations of discrete and continuous data for space and time.

Models

The following model classes are currently implemented in the package:

- `DiscreteSpaceModel` for discrete spatial data
- `ContinuousSpaceModel` for continuous spatial data
- `ContinuousSpaceDiscreteTimeModel` for continuous spatial and discrete temporal data
- `ContinuousSpaceContinuousTimeModel` for continuous spatial and continuous temporal data

Continuous models

Specifying continuous models require specifying an estimation mesh. Estimates are provided at the mesh nodes and estimates at the observation or prediction locations are interpolated from the node estimates. A higher number of nodes improves the estimation results, but increases computational time.

The following mesh classes are currently implemented in the package:

- `TemporalMesh` for temporal data
- `SpatialMesh` for spatial data
- `NonConvexHullMesh` for spatial data for creating a non-convex hull around the observations

The continuous spatial models assume that dependencies between the mesh nodes are specified as a function of distance with unknown scale and variance parameters to be estimated from the data. The function is of Matérn class, described in more detail in the reference.

Discrete models

Discrete models are specified with a neighborhood structure. For time, it is often assumed that the current time point t depends on the previous time point $t-1$ in some fashion. For example, the autoregressive model is of the form $y_t = \phi y_{t-1} + \epsilon$, where ϕ is the degree of dependency between subsequent observations and ϵ is the zero-centered Gaussian error term.

Regions are typically considered neighbors if they share the same border or are within certain radius from the centre points.

Please refer to the R-INLA documentation for more details of the autoregressive and the Besag models.

Installation

Test version of R-INLA is required to be installed first, see http://www.r-inla.org/download for the installation instructions. The `SpaceTimeModels` package is installed with the `devtools` package using the command `devtools::install_github("statguy/SpaceTimeModels")`. Additional packages are installed automatically from CRAN if needed. The package will be ready to use with the command

```
library(SpaceTimeModels)
```

Usage

The assembly line for constructing the models is the following:

1. Create model object.
2. Specify mesh for continuous spatial models / Specify the neighborhood structure for discrete spatial models.
3. Specify priors for space and time components. If omitted, default priors are used.
4. Specify covariates or intercept only (smoothing-only model).
5. Add

- Add observation locations, time points, observed responses and covariates
- Add validation locations, time points and covariates
- Add prediction locations, time points and covariates

6. Specify likelihood. If omitted, Gaussian likelihood is used as default.

Once the model is specified, the unknown parameters are ready to be estimated. Once the model is estimated, the results can be extract from the model object.

Continuous models

Model object Objects are created with the `new()` method (constructor), for example, the model object

```
model <- SpaceTimeModels::ContinuousSpaceDiscreteTimeModel$new()
```

Coordinates Due to numerical accuracy, spatial coordinates may need to be scaled down. For example, if the coordinates are such that $x = 6100000$ and $y = 50000$, 1000000 should be added to y to "match" with x. Furthermore, the coordinates should be scaled, e.g., by dividing by 1000000 and thus obtaining $x = 6.1$ and $y = 1.05$, which INLA handles better.

Spatial mesh creation Spatial mesh object is obtained, for example, with

```
mesh <- SpaceTimeModels::NonConvexHullMesh$new(knots = knots)
```

where `knots` provides coordinates for constructing the mesh. The `knots` object must be class of `SpatialPoints` from the `sp` package.

The mesh is specified with the `construct` method and depending on the type of the mesh, several parameters are needed to be supplied for the method that specify topology of the mesh. The parameters affect, for example, the number of nodes in the mesh. The mesh is found by a triangulation over the study area and the process should be completed rather quickly. However, the triangulation gets stuck sometimes and the INLA triangulation process or R has to be force-terminated (killed). A set of suitable mesh parameters is often found through iteration. It is adviced to run the models with s small number of mesh nodes first, especially with the spatio-temporal models, which may take very long time to estimate with large meshes.

The mesh parameters are listed in the help (which is obtained with R command `?x` where `x` is the class name, e.g. `?NonConvexHullMesh`) and explained in more detail in the reference, which the reader is adviced to go through.

The command `mesh$plot()` plots the mesh with the observation locations. Once created, the mesh is supplied for the model object with

```
model$setSpatialMesh(mesh)
```

Smoothing and covariate models Smoothing or intercept-only model can be specified with the `setSmoothingModel()` method, which includes only an intercept and a spatial or spatio-temporal random effect in the model. Such models provide smoothed observations filtered from noise occuring in space (and time). Smoothing can be accomplised with e.g. kernel estimators as well. However, spatial scale is usually left to be specified by hand for the unparametric methods.

To estimate the effect of covariates, the models provide the `setCovariatesModel` method, which takes the following arguments:

- `covariatesModel` specifying right-sided equation of the covariates to be included in the model.
- `covariates` specifying the covariates data frame.

For example

```
model$setCovariatesModel(covariatesModel = ~ a + b, covariates = covariates)
```

To view the full specification of the linear part of the model, use the `getLinearModel()` method. This will also print the random effect term as supplied to R-INLA. The part for the covariates differs from the specified for the categorial variables (factors) as they are replaced by corresponding dummy variables. The dummy variables appear also in the results.

Data stacks Observation, validation and prediction data is supplied in chunks that are tagged and stacked. Such design allows indexing the data so that the chunks can be later referenced by the tags. The chunks consist of coordinates, time indices (for spatio-temporal models), covariates (optional) and the observations for the observation chunk. The observations consist of responses and optionally offsets for count data. The following methods specify the data:

- `addObservationStack`
- `addValidationStack`
- `addPredictionStack`

For spatio-temporal data, each of the methods take in the `spdf` argument, which must be of class `STI` or `STIDF` from the `spacetime` package. For spatial data, the `sp` argument must be provided with an object of class `SpatialPoints` or `SpatialPointsDataFrame` from the `sp` package.

For example

```
model$addObservationStack(spdf=obs, response=obs@data$response, offset=obs@data$offset, covariates=obs@
model$addValidationStack(stdf=val, covariates=val@data, tag="val")
model$addPredictionStack(stdef=pred, covariates=pred@data, tag="pred")
```

TODO

Priors TODO

Discrete models

TODO

Priors TODO

Likelihood

The models support various types of response data such as continuous (Gaussian), binary (binomial) and count (Poisson). The response data type is specified with

```
model$setLikelihood("x")
```

where `x` is the selected likelihood. Please refer to the R-INLA documentation for all supported likelihoods. When using count data likelihoods such as binomial and Poisson, the offset term is supplied for the stack methods method via the `offset` argument. Otherwise the offset is assumed to equal to 1. See the section Data stacks.

TODO: offset argument for discrete models etc.

Estimation

Once the data and the model are specified, estimation is started with the `estimate()` method. Argument `verbose=TRUE` is recommended to be supplied to follow progress of the estimation. Note that the spatio-temporal models may take considerable amount of time and memory to be estimated.

Extracting results

The following methods provide basic summaries of the estimated models

- `summary()` for overall summary of the model.
- `summarySpatialParameters()` for summary of the spatial parameters.

To access the R-INLA result object directly, `getResult()` method is provided.

Model selection

Model selection can be performed with the same model object by respecifying the covariate model. However, for the continuous models the data stack needs to be reconstructed by first issuing the `clearStack()` method and then repeating the `add*Stack()` methods before estimation. The `summary()` method provides the WAIC measure for the model selection.

Examples

- Cameletti et al. (2012) reproduced with the `SpaceTimeModels` package, link.

References and other supporting material

- Lindgren, F., Rue, H. (2015). Bayesian Spatial Modelling with R-INLA. Journal of Statistical Software.
- Cameletti, M., Lindgren, F., Simpson, D., Rue, H. (2012). Spatio-temporal modeling of particulate matter concentration through the SPDE approach. AStA Advances in Statistical Analysis.
- Krainski, E.T., Lindgren, F., Simpson, D., Rue, H. The R-INLA tutorial on SPDE models.
- Gelman, A., Hwang, J., Vehtari, A. (2013). Understanding predictive information criteria for Bayesian models. Statistics and Computing.

Contact

Jussi Jousimo, jvj@iki.fi