Journal of Statistical Software
April 2023, Volume 106, Issue 11. doi: 10.18637/jss.v106.i11

disaggregation: An R Package for Bayesian Spatial Disaggregation Modeling

Anita K. Nandi
University of Oxford

Tim C. D. Lucas
University of Oxford

Rohan Arambepola
University of Oxford

Peter Gething
Telethon Kids Institute
Curtin University

Daniel J. Weiss
University of Oxford

Abstract

Disaggregation modeling, or downscaling, has become an important discipline in epidemiology. Surveillance data, aggregated over large regions, is becoming more common, leading to an increasing demand for modeling frameworks that can deal with this data to understand spatial patterns. Disaggregation regression models use response data aggregated over large heterogeneous regions to make predictions at fine-scale over the region by using fine-scale covariates to inform the heterogeneity. This paper presents the R package **disaggregation**, which provides functionality to streamline the process of running a disaggregation model for fine-scale predictions.

Keywords: Bayesian spatial modeling, disaggregation modeling, TMB.

1. Introduction

Methods for estimating high-resolution risk maps from aggregated response data over large spatial regions are becoming increasingly sought after in disease risk mapping (Li, Brown, Gesink, and Rue 2012; Diggle, Moraga, Rowlingson, and Taylor 2013; Wilson and Wakefield 2018), especially in malaria mapping (Sturrock *et al.* 2014; Sturrock, Bennett, Midekisa, Gosling, Gething, and Greenhouse 2016). Disaggregation regression, first applied in species distribution modeling in ecology (Keil, Belmaker, Wilson, Unitt, and Jetz 2013) has now become an important method in disease risk mapping (Weiss *et al.* 2019; Battle *et al.* 2019). The aggregation of response data over large heterogeneous regions is problematic for making fine-scale predictions, as relationships learned between variables at one spatial scale may not hold at other scales (Wakefield and Shaddick 2006). However, by using fine-scale information from related covariates we can inform the heterogeneity of the response variable of interest within the aggregated area.

Disaggregation modeling is unorthodox as the predictions are at a different scale to the response data, i.e., the number of rows in the covariate data is different to that of the response data. The spatial modeling software package, **INLA** (Rue, Martino, and Chopin 2009), or integrated nested Laplace approximation, has been shown to be very useful in a wide variety of circumstances, however it is not flexible enough for the unorthodox nature of the disaggregation problem except in the special case of the linear link function (Wilson and Wakefield 2018; Moraga, Cramb, Mengersen, and Pagano 2017). Disaggregation models can be implemented in **TMB** (Kristensen, Nielsen, Berg, Skaug, and Bell 2016), or template model builder, with a lot of flexibility, however the data manipulation required to format the model objects and construct the model definition in C++ (Stroustrup 2013) is non-trivial. The **disaggregation** package allows this process to be streamlined, to make it easy for the user to run disaggregation models at the expense of some flexibility.

2. Disaggregation modeling

Suppose we have response data, y_i, for N polygons, which corresponds to count data for the property of interest within that polygon. The process that is being counted occurs in continuous space that we model as a high-resolution, square lattice for convenience. The data, y_i, are assumed to be created by the aggregation of the counts over the polygon, i.e., the count data of the polygon is given by the sum of the count data for all the pixels within that polygon. An aggregation raster must be provided to transform from the pixel level predictions (rate) to count data, from which it is trivial to calculate the corresponding polygon value by summing up the values for the individual pixels within the polygon. For example, in epidemiology, we may have as our response the number of people that contract a certain disease in a given period of time (case incidence). Our rate would be the number of cases per population, where the aggregation raster corresponds to population. The case generating process is modeled at the pixel level, with these processes then aggregated to obtain the likelihood for the aggregated observed data.

For the disaggregation model we model the rate at pixel level, with the likelihood for the observed data given by aggregating these pixel level rates. The rate in pixel j of polygon i at location s_{ij} is given by:

$$\text{link}(\text{rate}_{ij}) = \beta_0 + \beta X_{ij} + \text{GP}(s_{ij}) + u_i \tag{1}$$

where β are regression coefficients, X_{ij} are covariate values, GP is a Gaussian random field and u_i is a polygon-specific iid effect. The user-defined link function is typically log, identity and logit for Poisson, normal and binomial likelihoods, respectively. The Gaussian random field has a Matérn covariance function, defined by:

$$C(d) = \frac{\sigma^2}{\Gamma(\nu)2^{\nu-1}}(\kappa d)^{\nu} K_{\nu}(\kappa d)$$

with two hyperparameters: $\rho = \frac{\sqrt{8\nu}}{\kappa}$, the range of the field (beyond which correlation is $<$ 0.1), and σ, the marginal standard deviation, relating to the magnitude of the variation in the field. The parameters ρ, σ and ν are very difficult to identify together, therefore we fix ν, to be able to identify ρ and σ. The parameter ν is the smoothness and is fixed at 1, as it is considered the more natural basic choice for two-dimensional ($d = 2$) models (Lindgren and Rue 2015). If the application requires a higher level of smoothness, this can also be controlled by larger values of ν, nevertheless it comes with a computational cost. The Gamma function is given by $\Gamma(\nu) = (\nu - 1)!$ and K_ν is the modified Bessel function of the second kind.

As we are working in a Bayesian setting, each of the model parameters and hyperparameters are given a prior, which is discussed later.

Incidence is modeled at the pixel level, and the response data exists as count data at the polygon level. Therefore, to calculate the likelihood we must aggregate the pixel rate, via the aggregation raster a_{ij}, to get the expected polygon count. This aggregation is defined by:

$$\text{cases}_i = \sum_{j=1}^{N_i} a_{ij}\text{rate}_{ij} \tag{2}$$

$$\text{rate}_i = \frac{\text{cases}_i}{\sum_{j=1}^{N_i} a_{ij}}$$

where N_i is the total number of pixels in polygon i. The different likelihoods correspond to slightly different models (y_i is the response count data):

- *Poisson*

$$y_i \mid \beta_0, \beta_i, \text{GP}, u_i \sim \text{Poisson}(\text{cases}_i)$$

- *Gaussian*

$$y_i \mid \beta_0, \beta_i, \text{GP}, u_i \sim \text{Normal}(\text{cases}_i, \sigma_i)$$

  Here $\sigma_i = \sigma\sqrt{\sum_j a_{ij}^2}$, where σ is the pixel-level dispersion (a parameter learnt by the model).

- *Binomial*

$$y_i \mid \beta_0, \beta_i, \text{GP}, u_i \sim \text{Binomial}(M_i, \text{rate}_i)$$

In the example of disease mapping, Poisson or Gaussian likelihoods could be used when the quantity observed, y_i, is the total number of cases in a given polygon. The binomial model could be used when y_i is the prevalence of a disease in a sample of M_i people in the polygon.

The pixel predictions of incident rate are calculated from the fitted model parameters using Equation 1.

2.1. Priors

For each of the model parameters and hyperparameters we specify priors. The regression parameters and intercept are given Gaussian priors, where the default priors are $\beta_0 \sim N(0, 2)$ and $\beta_i \sim N(0, 0.4)$. It is expected that some of the spatial variation can be described by the covariates, and the Gaussian field can help describe the remaining spatial variation that is missing from the covariate information. Therefore, the priors on the covariates can be set to allow the covariates to explain some, but not all, of the variation in the response data. For the Gaussian random field, penalized complexity priors are used, which are constructed to penalize against deviating from the simpler base model, which in this case is a flat field, i.e., zero variance and infinite range (Fuglstad, Simpson, Lindgren, and Rue 2018). A penalized complexity prior is placed on the scale and range parameters of the random field such that

$$\mathsf{P}(\rho < \rho_{\text{min}}) = \rho_{\text{prob}}$$
$$\mathsf{P}(\sigma > \sigma_{\text{max}}) = \sigma_{\text{prob}}$$

where the values ρ_{min}, ρ_{prob}, σ_{max}, σ_{prob} are set by the user. The default values for these parameters within the package are driven by the nature of the data provided in the model. The default prior for ρ_{min} is set at a third of the spatial area covered by the polygons, and ρ_{prob} is set at 0.1. The default prior for σ_{max} is set to the standard deviation of the normalized response data, and σ_{prob} is set at 0.1.

The joint penalized complexity prior, π, corresponding to a base model with infinite range and zero variance (Fuglstad *et al.* 2018) is given by:

$$\log\left(\pi(\sigma,\rho)\right) = \log\left(\tilde{\lambda}_1\right) + \log\left(\tilde{\lambda}_2\right) - 2\log(\rho) - \frac{\tilde{\lambda}_1}{\rho} - \tilde{\lambda}_2\sigma$$

where

$$\tilde{\lambda}_1 = -\rho_{\text{min}}\log(\rho_{\text{prob}}) \qquad \text{and} \qquad \tilde{\lambda}_2 = -\frac{\log(\sigma_{\text{prob}})}{\sigma_{\text{max}}}$$

This prior shrinks the field towards a base model with zero variance and infinite range, in other words regularizing towards a flatter field with smaller magnitude.

The polygon-specific effects $u_1, \ldots, u_N$ have Gaussian priors centered at 0 with standard deviation σ_u (where the precision $\tau_u = 1/\sigma_u^2$). A penalized complexity prior is placed on σ_u (Simpson, Rue, Riebler, Martins, and Sørbye 2017) such that

$$P(\sigma_u > \sigma_{u,\,\text{max}}) = \sigma_{u,\,\text{prob}}$$

where values $\sigma_{u,\,\text{max}}$ and $\sigma_{u,\,\text{prob}}$ are set by the user. The penalized complexity prior, π_u, corresponding to a base model with no polygon-specific effect (Simpson *et al.* 2017) is given by:

$$\log\left(\pi(\lambda)\right) = \log\left(\frac{\lambda}{2}\right) - \frac{3}{2}\log(\tau_u) - \frac{\lambda}{\sqrt{\tau_u}}$$

where

$$\lambda = -\frac{\log(\sigma_{u,\,\text{prob}})}{\sigma_{u,\,\text{max}}} \qquad \text{and} \qquad \tau_u = \frac{1}{\sigma_u^2}$$

This prior shrinks towards a base model of no polygon-specific effect.

For models that use a Gaussian likelihood, a log gamma prior is set on the log of the precision, $\log\tau_u \sim \log\Gamma(\text{shape} = 1, \text{rate} = 5 \times 10^{-5})$, to regularize the dispersion, σ_u, to take low values. This is chosen to be consistent with the prior set by **INLA** for the dispersion of the normal likelihood. We aim for consistency with **INLA** as **INLA** is one of the most commonly used packages for Bayesian spatial modeling and due to the parallels in computation approach such as the use of a Laplace approximation and the stochastic partial differential equation approximation to the Gaussian random field.

3. Implementation

The **disaggregation** package is built on template model builder (**TMB**, Kristensen *et al.* 2016), which is a tool for flexibly building complex models based on C++. **TMB** combines the packages **CppAD** (Bell 2012), for automatic differentiation in C++, Eigen (Guennebaud, Jacob, Avery, Bachrach, Barthelemy *et al.* 2021), a C++ library for linear algebra, and **CHOLMOD** (Chen, Davis, Hager, and Rajamanickam 2008), a package for efficient computation of sparse matrices. The use of these packages allows an efficient implementation of the automatic Laplace approximation (Skaug and Fournier 2006) with exact derivatives which gives an approximation to the Bayesian posterior. **TMB** calculates first and second order derivatives of the objective function using automatic differentiation (Griewank and Walther 2008).

TMB is a package which allows the user to define and fit latent variable models. The user defines the objective function (typically the likelihood or posterior density function) in C++ and **TMB** then implements the Laplace approximation to integrate out random effects and generates functions for evaluating the objective and gradient (via automatic differentiation). These functions can then be passed to an optimizer (for maximum likelihood estimation or to find the posterior maximum) or to a Markov chain Monte Carlo (MCMC) sampler in R (R Core Team 2023). **TMB** can fit both frequentist and Bayesian models. In this case we have chosen the Bayesian approach as it is most widely used in the spatial modeling community.

The **disaggregation** package contains a C++ function that defines the model and computes the joint likelihood as a function of the parameters and the random effects, in the format expected by **TMB**. The **TMB** package then calculates estimates of both parameters and random effects using a Laplace approximation. Evaluation of the objective function and its derivatives is performed via R.

3.1. Other implementations

The simplest method to disaggregate data from the polygon level to the high-resolution pixel level is to simply apply the value given by the polygon across all pixels within that polygon. This is the approach used by the `rasterize` function in the **raster** package, the `fasterize` function in the **fasterize** package and `st_interpolate_aw` from the **sf** package (Hijmans 2023; Ross 2022; Pebesma 2018). Similarly, for time-series data there are functions such as `td` in the **tempdisagg** package (Sax and Steiner 2023). These methods are not based on a statistical model and cannot help estimate the heterogeneous distribution of the disease within a polygon. Furthermore, these methods cannot make predictions outside of the area for which polygon data is available.

A number of other implementations use statistical models to estimate high resolution disease risk from aggregated disease data. To the best of our knowledge, these all only include covariates at the polygon level rather than at the pixel level. The popular R package **INLA** can perform disaggregation regression in the simplest case of a normal likelihood, an identity link function and no high resolution covariates (Moraga *et al.* 2017; Lindgren and Rue 2015; Wilson and Wakefield 2018). The **SDALGCP** package fits the model using Monte Carlo maximum likelihood (Johnson, Diggle, and Giorgi 2019). It can only fit models with a Poisson likelihood with covariates at the polygon level. Finally, the **lgcp** package uses a data augmentation approach to fit models to aggregated data (Taylor, Davies, Rowlingson, and Diggle 2013). This Monte Carlo data augmentation approach will be very slow for large areas or large numbers of cases.

In this paper we compare our method to MCMC. MCMC methods involve constructing Markov chains which (after a sufficient warm up period) produce samples from the desired probability distribution. In high dimensional problems, these methods can be extremely slow as many steps are needed to converge to and effectively sample from the target distribution. This is particularly problematic when computing the likelihood is costly, as the likelihood is typically evaluated at each step. In spatial settings with a Gaussian field, MCMC becomes infeasible for disaggregation modeling. Our method presented in the paper leverages the Laplace approximation to provide a much faster way of fitting disaggregation models.

4. Package usage

In this section we show how to use the **disaggregation** package using a dataset of aggregated malaria case counts across Madagascar in 2016. Malaria is an infectious disease caused by parasites of the Plasmodium group, transmitted by Anopheles mosquitoes. Malaria transmission is therefore closely related to mosquito and parasite development. Environmental factors such as temperature, rainfall and elevation have been shown to have significant effects on mosquito survival and development; in general mosquitoes favor warm, humid environments with moderate rainfall. Therefore, such environmental covariates would be useful in a malaria disaggregation model to inform fine-scale distributions. In this model we use the environmental covariates of mean land surface temperature (LST), elevation, and Enhanced Vegetation Index (EVI), at a resolution of approximately $5 \times 5\ km^2$, to inform spatial heterogeneity in malaria risk. These covariates are obtained from the Moderate Resolution Imaging Spectroradiometer (MODIS), which provides many measurements over the entire Earth's surface (`https://modis.gsfc.nasa.gov/data/`).

Malaria incidence rate is often given per thousand people per year given the term Annual Parasite Index (API). In this case our pixel predictions correspond to malaria incidence rate, so we use population to aggregate pixel incidence rate by summing the number of cases (rate weighted by population). Raster surfaces of population for the years 2010 and 2015 at a resolution of approximately $5 \times 5\ km^2$, were created using data from WorldPop (Tatem 2017) and from GPWv4 (NASA 2018) where WorldPop did not have values. The population raster for 2016 was created by linear interpolation between 2010 and 2015 and extending out to 2016. This interpolation method results in a small amount of uncertainty in population raster, however this is expected to be negligible compared to the model uncertainty.

Covariate rasters or population rasters may be accompanied by significant uncertainty. There is no robust way within the package to propagate this uncertainty. However, if the user were able to sample from the covariate rasters and run many instances of the model, they could estimate an overall uncertainty. Whether this is a good approximation of the true uncertainty will depend on the specific use case.

The covariate rasters and aggregation raster provided must be of the same spatial scale and must be spatially aligned. For spatially misaligned rasters, there are packages in R to align rasters such as `align_rasters` in the **gdalUtils** package Greenberg and Mattiuzzi (2022).

The latest version of **disaggregation** should always be available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=disaggregation`. Run the following commands to install and load the package.

```
R> install.packages("disaggregation")
```

```
R> library("disaggregation")
R> set.seed(5)
```

We then read in the data to use in the **disaggregation** package. The `shapefile` function and `raster` function are contained within the **raster** package, they are functions to read spatial data and raster data respectively. These functions return objects of class '`SpatialPolygonsDataFrame`' and '`RasterLayer`' respectively. '`SpatialPolygonsDataFrame`' is an R class that holds spatial data in the form of polygons with attributes, in this case the attributes are number of malaria cases within the polygon and polygon ID. A '`RasterLayer`' is an R class that holds a single raster. The function `getCovariateRasters` is a helper function contained within the **disaggregation** package. From a user defined directory name and shapefile, all the raster files within that directory are read and a '`RasterStack`' is created with the same extent as the shapefile provided. In this example we are building a '`RasterStack`' of the covariate rasters with the same extent as the population raster. A '`RasterStack`' is a collection of '`RasterLayer`' objects with the same spatial extent and resolution.

The main functions are `prepare_data`, `disag_model` and `predict`.

Firstly, we use the `prepare_data` function to setup all the data in the format needed in the disaggregation modeling. This function performs various data manipulation tasks to create objects that are necessary for fitting the model. The required input data for the `prepare_data` function are:

- `polygon_shapefile`: '`SpatialPolygonsDataFrame`' containing the response data. It must contain IDs and response data.
- `covariate_raster`: '`RasterStack`' of covariates to be used in the model.
- `aggregation_raster`: '`RasterLayer`' used as the weights to aggregate the pixel values within a polygon.
- `mesh.args`: List of parameters to control the mesh used for the Gaussian Field component.
- `id_var`: Variable name of the ID variable in the `polygon_shapefile`.
- `response_var`: Variable name of the response variable in the `polygon_shapefile`.
- `na.action`: Boolean, whether to deal with `NA`s or not.
- `ncores`: Number of cores to perform the parallel extraction over.

An optional aggregation raster can be provided. The aggregation raster defines how the pixels within each polygon are aggregated. The disaggregation model performs a weighted sum of the pixel predictions, weighted by the pixel values in the aggregation raster, as shown in Equation 2. In this case our pixel predictions are malaria incidence rate, so we use the population raster to aggregate pixel incidence rate by summing the number of cases (rate weighted by population). If no aggregation raster is provided a uniform distribution is assumed, i.e., the pixel predictions are aggregated to polygon level by summing the pixel values unaltered.

The values of the covariates (as well as the aggregation raster, if given) are extracted at each pixel within the polygons and stored as a data.frame with a row for each pixel and a column for each covariate (`parallelExtract` function). The extraction of each covariate is performed in parallel over the number of cores defined by the argument `ncores`. The values extracted from the aggregation raster are returned as an array of values, one for each pixel. In order to know which pixels (i.e., which rows) are contained in each polygon, a matrix is constructed that contains the start and end pixel index for each polygon (`getStartendindex` function).

To fit a model with a Gaussian field we approximate a Gaussian field using Gaussian Markov random fields (GMRF), and solve using the stochastic partial differential equation (SPDE) approach (Lindgren, Rue, and Lindström 2011). This approach requires building a mesh, i.e., splitting the space into finite elements, and calculating field values at the mesh nodes. If a mesh is too fine, the field values take too long to compute, however if the mesh is too coarse, the resulting field is a poor approximation and leads to mesh artifacts in the predictions. The 2D mesh for the spatial field is built using the `build_mesh` function, which makes use of the **INLA** function `inla.mesh.2d`. The user can control the parameters of the mesh, including the granularity, using the argument `mesh.args` in the `build_mesh` function. The parameters `max.edge`, `cut` and `offset` that can be set in the `mesh.args` list are defined within the `inla.mesh.2d` function. By providing two values to the `max.edge` parameter, the mesh contains an inner region of finer mesh, and an outer coarse region. This approach allows the region of interest to have a fine as mesh as necessary without also requiring time consuming computations in the sea regions. The mesh can take several minutes to construct, so to prepare the data without building the mesh the user can set the `makeMesh` flag to `FALSE`. However, it would then not be possible to fit the disaggregation model without the mesh.

If there are any `NA`s in the response or covariate data within the polygons the `prepare_data` method will return an error. This can be dealt with using the `na.action` flag, which is automatically off. Ideally the `NA`s in the data would be dealt with by the user beforehand, however, setting `na.action = TRUE` will automatically deal with `NA`s. It removes any polygons that have `NA`s as a response, sets any aggregation pixels with `NA` to zero and sets covariate `NA` pixels to the median value for that covariate across all polygons.

```
R> dis_data <- prepare_data(polygon_shapefile = shapes,
+    covariate_rasters = covariate_stack,
+    aggregation_raster = population_raster,
+    mesh.args = list(max.edge = c(0.7, 8), cut = 0.05, offset = c(1, 2)),
+    id_var = "ID_2", response_var = "inc", na.action = TRUE, ncores = 8)
```

We can see a summary of the data, using the generic `summary` function, and `plot` the data. The summary function returns information on how many pixels and polygons the data contains, how many pixels in the smallest and largest polygons and a summary of the covariate data. The `plot` functions plots a map of the polygon response data, the covariate rasters and the **INLA** mesh, as shown in Figure 1.

```
R> summary(dis_data)

They data contains 109 polygons and 28892 pixels
The largest polygon contains 867 pixels and the smallest polygon contains
 1 pixels
There are 3 covariates
```

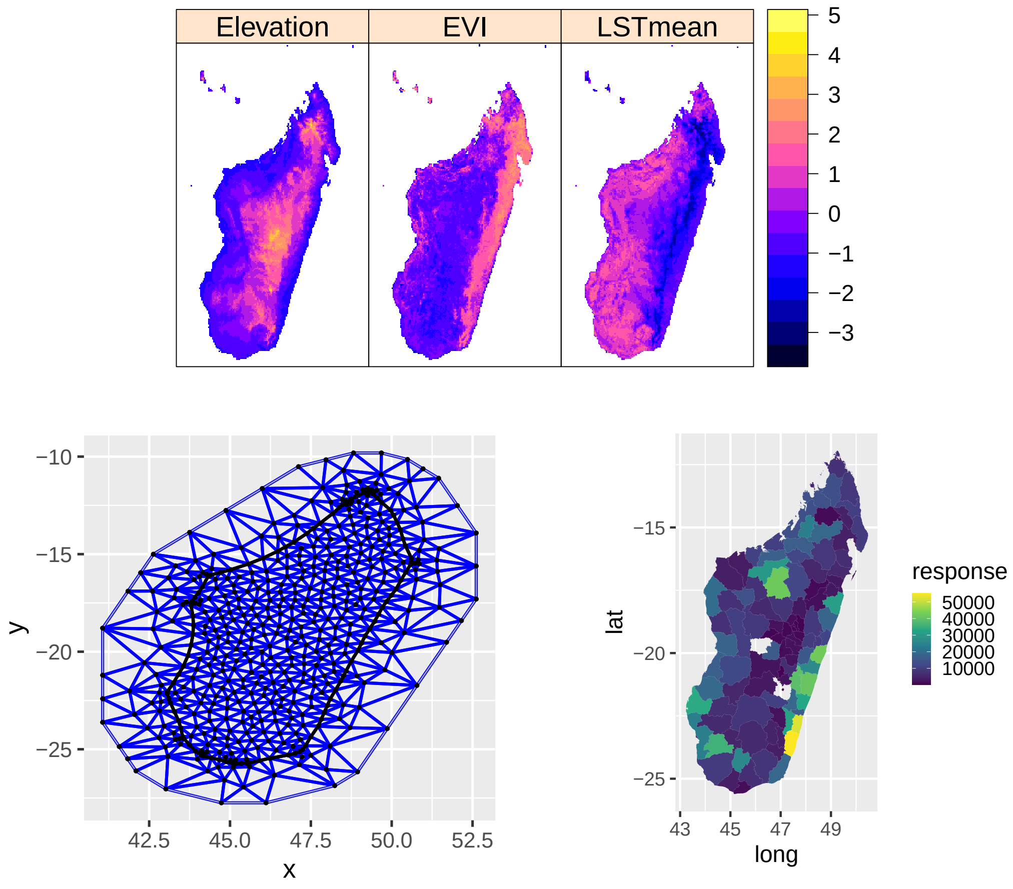

Figure 1: Maps of Madagascar showing the data used in the disaggregation model. These plots are produced when calling the `plot` function on the `disag_data` object. The plots show maps of the four covariates used in the model (top), the number of malaria cases in each administrative unit (bottom right), and `inla.mesh` object that will be used to make the spatial field (bottom left).

```
Covariate summary:
Elevation                EVI                LSTmean
Min.   :-1.174670   Min.   :-2.70028   Min.   :-3.33993
1st Qu.:-0.848255   1st Qu.:-0.73278   1st Qu.:-0.73083
Median :-0.257381   Median :-0.36756   Median : 0.23743
Mean   : 0.009915   Mean   : 0.01077   Mean   : 0.01572
3rd Qu.: 0.713218   3rd Qu.: 0.59588   3rd Qu.: 0.83374
Max.   : 4.585433   Max.   : 3.14432   Max.   : 2.09070
```

```
R> plot(dis_data)
```

The `prepare_data` function returns an object of class ‘`disag_data`’, which is designed to be used directly in the `disag_model` function.

Now we can fit the disaggregation model using `disag_model`. The required inputs for the `disag_model` function are:

- `data`: Object of class '`disag_data`' returned by the `prepare_data` function.
- `priors`: List of priors and hyperpriors to use for the model. For any not set, the default priors will be used.
- `iterations`: Maximum number of iterations the model can run for to find an optimal point.
- `family`: Likelihood function. Options are `gaussian`, `poisson` and `binomial`.
- `link`: Link function used in the model. Options are `logit`, `log` and `identity`. This would typically be log, identity and logit for Poisson, normal and binomial likelihoods, respectively.

Here we use a Poisson likelihood for the incidence count data with a log link function. The spatial field and iid effect are components of the model by default, they can be turned off using the `field` and `iid` flags. In this example we use the default priors. Hyperpriors for the field are implemented as penalized complexity priors – specify ρ_{min} (`prior_rho_min`) and ρ_{prob} (`prior_rho_prob`) for the range of the field, where $P(\rho < \rho_{\text{min}}) = \rho_{\text{prob}}$, and σ_{max} (`prior_sigma_max`) and σ_{prob} (`prior_sigma_prob`) for the variation of the field, where $P(\sigma > \sigma_{\text{max}}) = \sigma_{\text{prob}}$. Similarly, penalized complexity priors are implemented for the iid effect (`prior_iideffect_sd_max` and `prior_iideffect_sd_prob`). Any of the priors and hyperpriors can be set in the list of priors given to the `priors` argument of the `disag_model` function. In order to print more verbose output the user can set the `silent` argument to `FALSE`.

```
R> fitted_model <- disag_model(data = dis_data,
+    iterations = 1000, family = "poisson", link = "log")
```

We can get a summary and plot of the model output. The `summary` function gives the estimate and standard error of the fixed effect parameters in the model, as well as the negative log likelihood and in-sample performance metrics (root mean squared error, mean absolute error, Pearson correlation coefficient and Spearman rank correlation coefficient). The standard error is calculated based on the approximate posterior from the **TMB** package. The `plot` function produces two plots: one of the fixed effects parameters and one of the observed data against in-sample predictions, as shown in Figure 2.

```
R> summary(fitted_model)

Likelihood function: poisson
Link function: log
Model parameters:
Estimate Std. Error
intercept         -3.1840853  0.2656827
slope             -0.4583419  0.1993289
slope              0.3883232  0.2221936
slope              0.1925308  0.2918306
iideffect_log_tau  1.0881520  0.2770572
log_sigma          0.0253808  0.1583851
log_rho            0.5678399  0.3199920
```

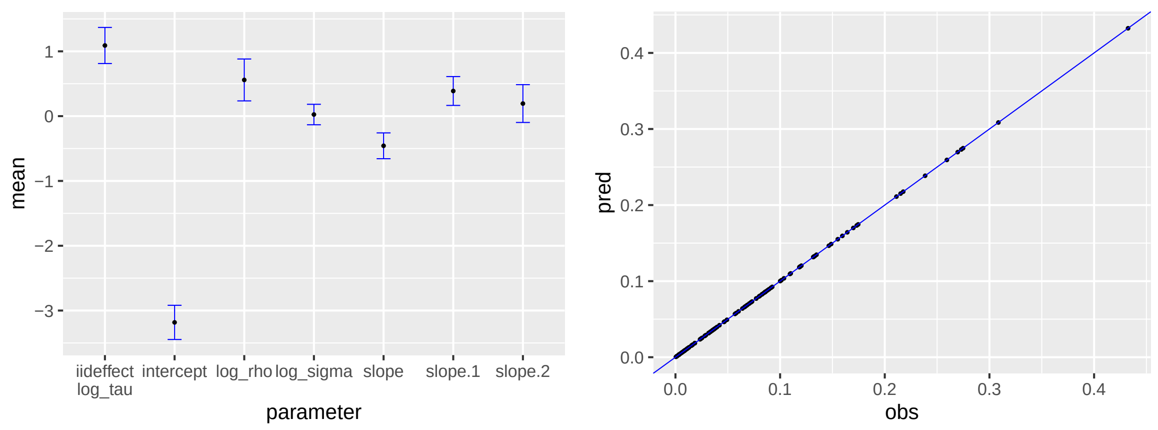

Figure 2: Plot summarizing the results of the fitted model. These plots are produced when calling the `plot` function on the `disag_model` object. The fixed effects plot (left) shows the fitted parameter values with uncertainty estimation for all the fixed effects in the model. The in-sample performance plot (right) shows the predicted incidence rate values for each polygon in the data against the observed values for that polygon in the data.

```
Model convergence: 0 (both X-convergence and relative convergence (4))
Negative log likelihood:  977.440180015428

In sample performance:
RMSE        MAE pearson spearman log_pearson
1 1.317422 0.9918037       1        1   0.9999994
```

```
R> plot(fitted_model)
```

The `disag_model` function returns an object of class '`disag_model`', which is designed to be used directly in the `predict` function. Therefore, now that we have fitted the model, we are ready to predict the malaria incidence rate across Madagascar.

To predict over a different spatial extent to that used in the model, a '`RasterStack`' covering the region to make predictions over can be passed as the `newdata` argument. If this argument is not given, predictions are made over the covariate rasters used in the fit. If the user wants to include the iid effect from the model as a component in the prediction then the `predict_iid` logical flag should be set to `TRUE`, otherwise, the iid effect will not be predicted.

For the uncertainty calculations, parameter values are sampled from the posterior distribution and summarized. The number of parameter draws used to calculate the uncertainty is set by the user via the `N` parameter (default: 100), and the size of the credible interval (e.g., 75%, 95%) to be calculated when summarising is set via the argument `CI` (default: 0.95).

```
R> model_prediction <- predict(fitted_model)
```

The function `predict` returns a object of class '`disag_prediction`' containing a list of two objects: the mean predictions and the uncertainty rasters. The mean predictions contain a raster of the mean prediction of the incidence rate, as well as rasters of the field, iid (if

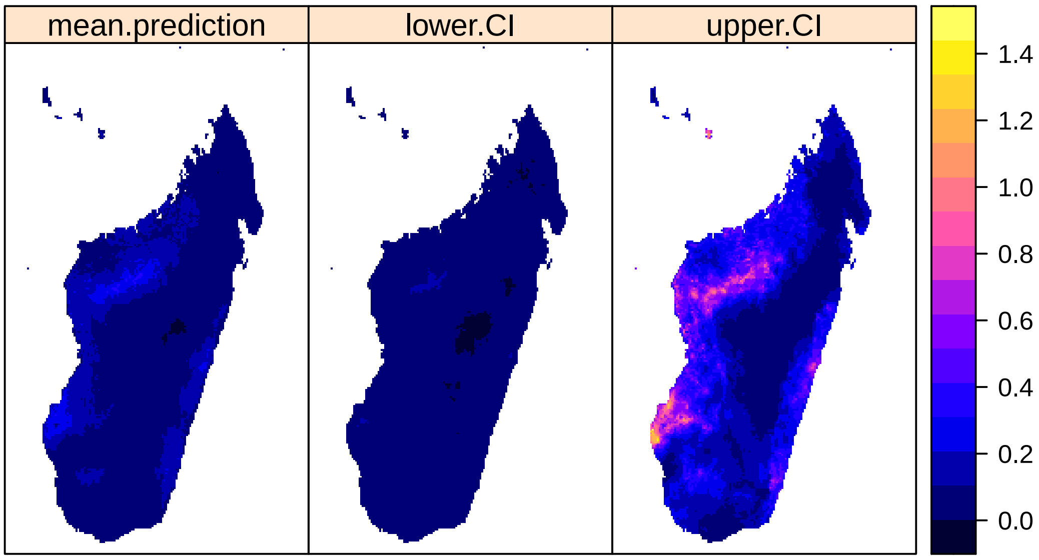

Figure 3: Maps of Madagascar showing the fine-scale predictions of mean, lower (2.5%) and upper (97.5%) credible intervals of the malaria incidence rate from the disaggregation model. These maps are produced when calling the `plot` function on the `disag_prediction` object.

predicted) and covariate component of the linear predictor. The uncertainty predictions contains a '`RasterStack`' of the prediction realizations and a '`RasterStack`' of the upper and lower credible intervals.

The `plot` function can be used on the `disag_prediction` objects, as shown in Figure 3. From Figure 1, it can be seen that the polygon response data does not include the islands, whereas the covariate rasters do. As can be seen in Figure 3, out-of-sample malaria predictions of these islands have been made, however these predictions do not contribute to the in-sample performance in Figure 2, nor to cross validation performance. In fact, these islands are not part of Madagascar at all, they are the islands of Comoros and Mayotte.

```
R> plot(model_prediction)
```

Using three simple functions within the **disaggregation** package we have been able to fit a Bayesian spatial disaggregation model and predict pixel-level incidence rate across Madagascar using aggregated incidence data and pixel-level environmental covariates.

The same technique can be used for spatio-temporal disaggregation modeling. In order to achieve this, dynamic covariates are required, for example, annual covariate rasters, as well as a spatio-temporal field, which could be achieved by adding a time varying autoregressive component to the spatial field. A difficulty with this approach is that it results in significant increase in computational complexity of the optimization problem, most notably from the spatio-temporal field, leading to a significant increase in the time taken to perform the optimization. The package can be developed to include spatio-temporal disaggregation models, however that is beyond the scope of this paper.

5. Comparison with Markov chain Monte Carlo (MCMC)

In this section we show performance comparisons between modeling using the Laplace approximation provided by the **disaggregation** package, based on **TMB**, and using MCMC. Given a function to evaluate the probability density of a distribution at any given point in parameter space, MCMC algorithms construct Markov chains to generate samples from this distribution. These algorithms are often slow, particularly in high-dimensional settings, as it can take a long time to converge to and effectively sample from the stationary distribution. The density is evaluated at each step, so this problem is compounded when evaluating this density is computationally expensive. In contrast, the **disaggregation** package approximates using a Laplace approximation to the posterior to generate posterior samples. This only requires the posterior to be maximized to find the posterior mode and therefore involves relatively few evaluations of the posterior density compared to MCMC techniques, although potentially at the expense of less accurate posterior samples. Here we compare the time and performance of the two techniques. Note that slightly different results might be obtained depending on the specific computing environment, in particular the linear algebra library used.

The model described in Section 4 for malaria in Madagascar has been optimized using the **disaggregation** package. Here we fit the same model by running MCMC using the **tmbstan** package (Monnahan and Kristensen 2018), with the NUTS (no-U-turn sampler) algorithm (Hoffman and Gelman 2014) using four chains. It is important to note that this is a useful feature of the **disaggregation** package, to be able to create the **TMB** model object (using the `disag_model` function with one iteration) and pass it directly to **tmbstan**. The model is fitted by running the MCMC algorithm for 8000 iterations with 2000 of those as warm-up, which took 56 hours. This number of iterations was chosen by running the MCMC algorithm repeatedly, starting at 1000 iterations, doubling the number of iterations each time until the value of the MCMC convergence statistic, $\hat{R}$, dropped below 1.05 for all model parameters. The property $\hat{R}$ is known as the Gelman-Rubin diagnostic (Gelman and Rubin 1992), and is commonly used to evaluate MCMC convergence across multiple chains. This is achieved by comparing the estimated between-chains and within-chain variances for each model parameter. Non-convergence is indicated by large differences between the variances. In contrast, fitting the model using the Laplace approximation via **TMB** within the **disaggregation** package took 85 seconds.

	MCMC		TMB	
Parameter	Mean	SD	Mean	SD
Intercept	−3.26	0.47	−3.18	0.27
Slope 1	−0.45	0.21	−0.46	0.20
Slope 2	0.41	0.23	0.39	0.22
Slope 3	0.18	0.30	0.19	0.29
$\log(\tau_u)$	1.02	0.28	1.09	0.28
$\log(\sigma)$	0.12	0.19	0.03	0.16
$\log(\rho)$	0.80	0.38	0.57	0.32

Table 1: Fitted model parameter values using both MCMC (56 hours) and using **TMB** (85 seconds) within the **disaggregation** package.

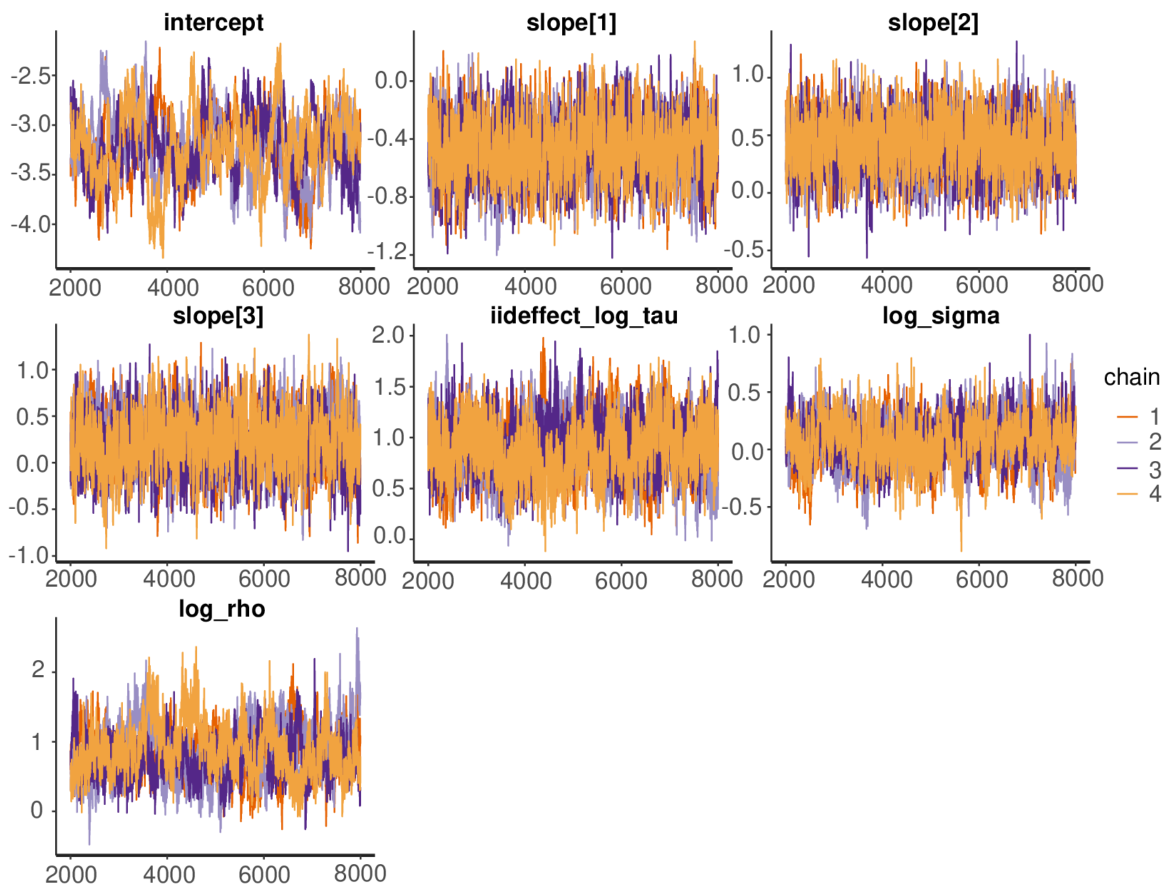

Figure 4: Trace of the fixed effects parameters for MCMC using NUTS sampling, running the algorithm for 56 hours. It was run for 8000 iterations with 2000 of those as warmup.

Fitted parameter values for both of these methods are given in Table 1. It can be seen that the model parameters are very similar between the two methods. It is worth noting that the field parameters given in Table 1 are $\log(\rho)$ and $\log(\sigma)$. If we transform these parameters to their natural form, the values given in Table 1 correspond to a difference in the range of the field, ρ, from 1.76 ± 0.56 (TMB) to 2.23 ± 0.84 (MCMC), and a difference in the standard deviation of the field, σ, from 1.03 ± 0.16 (TMB) and 1.13 ± 0.22 (MCMC). Considering these values, along with their large accompanying uncertainties, these field predictions are very similar between the two methods. In general, the Laplace approximation will perform worse when the posterior is less Gaussian; this occurs when the prior has more influence in the model, for example, when there is less data or for hierarchical parameters.

```
R> library("tmbstan")
R> model_object <- make_model_object(data = dis_data,
+    family = "poisson", link = "log")
R> start <- Sys.time()
R> mcmc_out <- tmbstan(model_object, chains = 4, iter = 8000, warmup = 2000,
+    cores = getOption("mc.cores", 4))
R> end <- Sys.time()
```

```
R> print(end - start)
R> stan_trace(mcmc_out, pars = c("intercept", "slope[1]", "slope[2]",
+    "slope[3]", "iideffect_log_tau", "log_sigma", "log_rho"))
R> summary(mcmc_out)$summary[c("intercept", "slope[1]", "slope[2]",
+    "slope[3]", "iideffect_log_tau", "log_sigma", "log_rho"), ]
```

The trace of the MCMC parameter values is given in Figure 4. It can be seen that the MCMC algorithm has been run for long enough to get sufficient chain mixing. The **disaggregation** package produces similar results to the MCMC algorithm. However the model fitting using the **disaggregation** package took 85 seconds in contrast to the 56 hours taken for the MCMC run. Therefore, it can be seen that the **disaggregation** package provides a quick and simple way to run disaggregation models, that can be prohibitively slow using MCMC.

6. Conclusions

Disaggregation modeling, which involves fitting models at fine-scale resolution using areal data over heterogeneous regions, has become widely used in fields such as epidemiology and ecology. The **disaggregation** package implements Bayesian spatial disaggregation modeling with a simple, easy to use R interface. The package includes simple data preparation, fitting and prediction functions that allow some user-defined model flexibility. In this paper we have presented an application of the package, predicting malaria incidence rate across Madagascar from aggregated count data and environmental covariates.

The modeling framework is implemented using the Laplace approximation and automatic differentiation within the **TMB** package. This allows fast, optimized calculations in C++. These disaggregation models are computationally intensive and take a long time using MCMC optimization techniques. Using **TMB**, the models are much faster and produce similar results.

Future work could be done extending the **disaggregation** package to include spatio-temporal disaggregation models. This would require a spatio-temporal field as well as dynamic covariates, and would be significantly more computationally intensive. Additionally, tools for cross-validation could be included within the package. Cross-validation of spatial models is non-trivial due to the spatial autocorrelation in the data.

The **disaggregation** package provides a simple, useful interface to perform spatial disaggregation modeling, with reasonable flexibility, as well as having the scope to be extended to more complex disaggregation models.

Computational details

The results in this paper were obtained using R 4.2.3 with the **TMB** 1.9.3 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/, apart from **INLA**, which can be installed in R using the command:

```
R> install.packages("INLA", repos = c(getOption("repos"),
+    INLA = "https://inla.r-inla-download.org/R/stable"),
+    dependencies = TRUE)
```

Data availability

All data used in this paper, including the results of the MCMC algorithm, are available in the supplementary materials on the journal web page.

Acknowledgments

We would like to thank the Bill and Melinda Gates Foundation for funding this research.

References

Battle KE, Lucas TCD, Nguyen M, Howes RE, Nandi AK, Twohig KA, Pfeffer DA, Cameron E, Rao PC, Casey D, Gibson HS, Rozier JA, Dalrymple U, Keddie SH, Collins EL, Harris JR, Guerra CA, Thorn MP, Bisanzio D, Fullman N, Huynh CK, Kulikoff X, Kutz MJ, Lopez AD, Mokdad AH, Naghavi M, Nguyen G, Shackelford KA, Vos T, Wang H, Lim SS, Murray CJL, Price RN, Baird JK, Smith DL, Bhatt S, Weiss DJ, Hay SI, Gething PW (2019). "Mapping the Global Endemicity and Clinical Burden of Plasmodium vivax, 2000–17: A Spatial and Temporal Modelling Study." *The Lancet*, **394**. doi:10.1016/S0140-6736(19)31096-7.

Bell BM (2012). "**CppAD**: A Package for C++ Algorithmic Differentiation." *Computational Infrastructure for Operations Research*, **57**, 10. URL https://cppad.readthedocs.io/en/latest/user_guide.html.

Chen Y, Davis TA, Hager WW, Rajamanickam S (2008). "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate." *ACM Transactions on Mathematical Software*, **35**(3), 22. doi:10.1145/1391989.1391995.

Diggle PJ, Moraga P, Rowlingson B, Taylor BM (2013). "Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm." *Statistical Science*, **28**(4), 542–563. doi:10.1214/13-sts441.

Fuglstad GA, Simpson D, Lindgren F, Rue H (2018). "Constructing Priors That Penalize the Complexity of Gaussian Random Fields." *Journal of the American Statistical Association*, pp. 1–8. doi:10.1080/01621459.2017.1415907.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4), 457–472. doi:10.1214/ss/1177011136.

Greenberg JA, Mattiuzzi M (2022). ***gdalUtils**: Wrappers for the Geospatial Data Abstraction Library (GDAL) Utilities.* R package version 2.0.3.2, URL https://CRAN.R-project.org/package=gdalUtils.

Griewank A, Walther A (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 105. 2nd edition. Siam.

Guennebaud G, Jacob B, Avery P, Bachrach A, Barthelemy S, *et al.* (2021). "**Eigen**." C++ package version 3.4.0, URL https://eigen.tuxfamily.org/.

Hijmans RJ (2023). ***raster**: Geographic Data Analysis and Modeling.* R package version 3.6-20, URL https://CRAN.R-project.org/package=raster.

Hoffman MD, Gelman A (2014). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research*, **15**(1), 1593–1623. URL https://jmlr.org/papers/v15/hoffman14a.html.

Johnson O, Diggle P, Giorgi E (2019). "A Spatially Discrete Approximation to Log-Gaussian Cox Processes for Modelling Aggregated Disease Count Data." *Statistics in Medicine*, **38**(24), 4871–4887. doi:10.1002/sim.8339.

Keil P, Belmaker J, Wilson AM, Unitt P, Jetz W (2013). "Downscaling of Species Distribution Models: A Hierarchical Approach." *Methods in Ecology and Evolution*, **4**(1), 82–94. doi:10.1111/j.2041-210x.2012.00264.x.

Kristensen K, Nielsen A, Berg CW, Skaug H, Bell BM (2016). "**TMB**: Automatic Differentiation and Laplace Approximation." *Journal of Statistical Software*, **70**(5), 1–21. doi:10.18637/jss.v070.i05.

Li Y, Brown P, Gesink DC, Rue H (2012). "Log Gaussian Cox Processes and Spatially Aggregated Disease Incidence Data." *Statistical Methods in Medical Research*, **21**(5), 479–507. doi:10.1177/0962280212446326.

Lindgren F, Rue H (2015). "Bayesian Spatial Modelling with R-**INLA**." *Journal of Statistical Software*, **63**(19), 1–25. doi:10.18637/jss.v063.i19.

Lindgren F, Rue H, Lindström J (2011). "An Explicit Link between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach." *Journal of the Royal Statistical Society B*, **73**(4), 423–498. doi:10.1111/j.1467-9868.2011.00777.x.

Monnahan C, Kristensen K (2018). "No-U-Turn Sampling for Fast Bayesian Inference in **ADMB** and **TMB**: Introducing the **adnuts** and **tmbstan** R packages." *PLOS One*, **13**(5). doi:10.1371/journal.pone.0197954.

Moraga P, Cramb SM, Mengersen KL, Pagano M (2017). "A Geostatistical Model for Combined Analysis of Point-Level and Area-Level Data Using **INLA** and **SPDE**." *Spatial Statistics*, **21**, 27–41. doi:10.1016/j.spasta.2017.04.006.

NASA (2018). "Gridded Population of the World (GPW), V4." URL http://sedac.ciesin.columbia.edu/data/collection/gpw-v4, accessed 2023-03-14.

Pebesma E (2018). "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal*, **10**(1), 439–446. doi:10.32614/rj-2018-009.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ross N (2022). ***fasterize**: Fast Polygon to Raster Conversion.* R package version 1.0.4, URL https://CRAN.R-project.org/package=fasterize.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392. doi:10.1111/j.1467-9868.2008.00700.x.

Sax C, Steiner P (2023). ***tempdisagg**: Methods for Temporal Disaggregation and Interpolation of Time Series.* R package version 1.1, URL https://CRAN.R-project.org/package=tempdisagg.

Simpson D, Rue H, Riebler A, Martins TG, Sørbye SH (2017). "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors." *Statistical Science*, **32**(1), 1–28. doi:10.1214/16-sts576.

Skaug HJ, Fournier DA (2006). "Automatic Approximation of the Marginal Likelihood in Non-Gaussian Hierarchical Models." *Computational Statistics & Data Analysis*, **51**(2), 699–709. doi:10.1016/j.csda.2006.03.005.

Stroustrup B (2013). *The C++ Programming Language*. 4th edition. Addison-Wesley.

Sturrock HJ, Bennett AF, Midekisa A, Gosling RD, Gething PW, Greenhouse B (2016). "Mapping Malaria Risk in Low Transmission Settings: Challenges and Opportunities." *Trends in Parasitology*, **32**(8), 635–645. doi:10.1016/j.pt.2016.05.001.

Sturrock HJW, Cohen JM, Keil P, Tatem AJ, Le Menach A, Ntshalintshali NE, Hsiang MS, Gosling RD (2014). "Fine-Scale Malaria Risk Mapping from Routine Aggregated Case Data." *Malaria Journal*, **13**(1), 421. doi:10.1186/1475-2875-13-421.

Tatem AJ (2017). "WorldPop, Open Data for Spatial Demography." *Scientific Data*, **4**(170004), 2052–4463. doi:10.1038/sdata.2017.4.

Taylor BM, Davies TM, Rowlingson BS, Diggle PJ (2013). "**lgcp**: An R Package for Inference with Spatial and Spatio-Temporal Log-Gaussian Cox Processes." *Journal of Statistical Software*, **52**(4), 1–40. doi:10.18637/jss.v052.i04.

Wakefield J, Shaddick G (2006). "Health-Exposure Modeling and the Ecological Fallacy." *Biostatistics*, **7**(3), 438–455. doi:10.1093/biostatistics/kxj017.

Weiss DJ, Lucas TCD, Nguyen M, Nandi AK, Bisanzio D, Battle KE, Cameron E, Twohig KA, Pfeffer DA, Rozier JA, Gibson HS, Rao PC, Casey D, Bertozzi-Villa A, Collins EL, Dalrymple U, Gray N, Harris JR, Howes RE, Kang SY, Keddie SH, May D, Rumisha S, Thorn MP, Barber R, Fullman N, Huynh CK, Kulikoff X, Kutz MJ, Lopez AD, Mokdad AH, Naghavi M, Nguyen G, Shackelford KA, Vos T, Wang H, Smith DL, Lim SS, Murray CJL, Bhatt S (2019). "Mapping the Global Prevalence, Incidence, and Mortality of Plasmodium falciparum, 2000–17: A Spatial and Temporal Modelling Study." *The Lancet*, **394**. doi:10.1016/S0140-6736(19)31097-9.

Wilson K, Wakefield J (2018). "Pointless Spatial Modeling." *Biostatistics*, **21**(2), e17–e32. ISSN 1465-4644. doi:10.1093/biostatistics/kxy041.

Affiliation:

Anita Nandi
Malaria Atlas Project
Big Data Institute
University of Oxford
Old Road Campus
Roosevelt Dr, Oxford OX3 7DQ, United Kingdom
E-mail: anita.k.nandi@gmail.com

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/

April 2023, Volume 106, Issue 11 *Submitted:* 2019-12-11
doi:10.18637/jss.v106.i11 *Accepted:* 2022-07-23