The Journal of Open Source Software

RandomForestsGLS: An R package for Random Forests for dependent data

Arkajyoti Saha[1], **Sumanta Basu**[2], **and Abhirup Datta**[3]

1 Departments of Statistics, University of Washington **2** Department of Statistics and Data Science, Cornell University **3** Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health

DOI: 10.21105/joss.03780

Software

- Review
- Repository
- Archive

Editor: Fabian Scheipl

Reviewers:

- @mnwright
- @pdwaggoner

Submitted: 02 September 2021
Published: 06 December 2021

Summary

With the modern advancements in geographical information systems, remote sensing technologies, low-cost sensors, we are increasingly encountering massive datasets indexed by geo-locations and time-stamps. Modeling such high throughput spatio-temporal data needs to carefully account for spatial/serial dependence in the data, i.e., model the structures (patterns) of the data along space or time. A general model for describing dependent observations $(y_1, y_2, \cdots, y_n)$ and covariates $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ can be formulated as $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $f(\mathbf{x}_i)$ denotes the mean component and ϵ_i is the residual error, which accounts for the dependency in the data, beyond what is explained by the covariates. In this article, we work with second order dependency, i.e. we assume that the dependence is captured through the covariance function of the correlated stochastic process ϵ_i. Since the data are indexed by locations (spatial data) or time-points (temporal data), the correlation is a function of the "spatial distance" or "time-lag" between two observations.

The primary purpose of the package `RandomForestsGLS` is to fit nonlinear regression for spatial and temporal data. This package performs estimation through a novel rendition of Random Forests (RF); namely, RF-GLS, which makes use of the dependence structure in the data. With increasing computing capacity of personal computers, non-linear Machine Learning (ML) methods have become increasingly commonplace for regression and classification tasks due to their ability of capture complex interactions among the variables, that cannot be modeled by linear regression. However, many modern ML software lack the capability to efficiently account for the dependence structure in the data which leads to sub-optimal estimation. On the other hand, specialized software for spatial/temporal data are capable of properly modeling data correlation, but usually assume the relationships between the response and the covariates. `RandomForestsGLS` brings the best of both worlds together by bridging the gap between these two approaches by explicitly modeling the spatial/serial data correlation in the RF fitting procedure to substantially improve estimation of the regression function. In ML workflow, `RandomForestsGLS` can substitute existing ML methods in model training to take care of the dependence structure. Similarly, for traditional spatial modeling frameworks, `RandomForestsGLS` can be used instead of linear model based methods to account for nonlinearity. Additionally, `RandomForestsGLS` seamlessly leverages kriging to perform predictions at new locations for geo-spatial data, a primary objective in many spatial analysis.

Statement of need

RF is an ensemble of regression trees built on subsamples of the data. Regression trees iteratively partition the covariate space by greedily optimizing a split criterion, that minimizes the sum of intra-node variances. This split criterion can also be expressed as an Ordinary

Least-Square (OLS) loss with design matrix corresponding to the leaf nodes. OLS loss used in split criterion and node representative update rule in RF does not account for the dependence in the data, hence is not optimal. Additionally, responses are resampled in the "bagging" (bootstrap aggregation) (Breiman, 1996) step in RF, which, under dependence setting violates the primary requirement of bootstrapping that the data are independent and identically distributed. RF-GLS, implemented in `RandomForestsGLS`, mitigates the issue by using a dependency adjusted split criterion which incorporates the covariance structure in determining the optimal partition as well as the leaf node representatives. Additionally, RF-GLS facilitates resampling decorrelated contrasts, which is suitable for bootstrap sampling (Saha et al., 2021). The dependency is modeled using GP in a mixed model framework, allowing both estimation and prediction.

`RandomForestsGLS` is a statistical machine learning oriented R package for fitting RF-GLS on dependent data. `R` being a free software available under GNU General Public License with tailor-made statistical operators and vast library of statistical software, is very popular in the statistics community. The RF-GLS algorithm described in (Saha et al., 2021) involves computationally intensive linear algebra operations in nested loops which are especially slow in an interpreted language like `R`. In order to optimize the implementation of the algorithm, it becomes a necessity to use low-level languages which have a significantly steeper learning curve than that of `R` and lacks a majority of the inbuilt statistical operators in `R`. `RandomForestsGLS` brings the best of both worlds together by optimizing the implementation of the computationally challenging RF-GLS algorithm in the background with an easy to use `R` user interface.

`RandomForestsGLS` focuses on fast, parallelizable implementations of RF-GLS for spatial and time series data, which includes popular choices for covariance functions for both spatial (Matérn GP) and time series (autoregressive) model. The package is primarily designed to be used by researchers associated with the fields of statistical machine learning, spatial statistics, time series analysis, and scientific applications of them. A significant part of the code has already been used in Saha et al. (2021). With the combination of speed and ease-of-use that `RandomForestsGLS` brings to the table regarding non-linear regression analysis in dependent data, we hope to see this package being used in a plethora of future scientific and methodological explorations.

State of the field

There are a number of `R` packages that provides the option for implementing classical RF. Most notable of them being randomForest, which implements Breiman's Random Forests (Breiman, 2001) for Classification and Regression using the Fortran original code by Leo Breiman and Adele Cutler. Some of the other packages are xgboost, randomForestSRC, ranger, Rborist. For a detailed overview of the we refer the reader to CRAN Task View: Machine Learning & Statistical Learning. To the best of our knowledge, none of these packages explicitly account for model correlation, common in spatial and time-series data.

While the classical RF has also been used for a number of geo-spatial applications (see Saha et al. (2021) for references), most of them do not make methodological adjustments in RF to account for the spatial correlation in the data (CRAN Task View: Analysis of Spatial Data). Two recent works attempt to explicitly use spatial information in RF. Hengl et al. (2018) adds all pairwise spatial-distances as additional covariates. This does not need a dedicated package as it uses the classical RF. A detailed tutorial to implement this in `R` is available in GeoMLA. The other approach, Georganos et al. (2019) proposes geographically local estimation of RF. Though this approach also employs the classical RF implemented in `randomForest`, it is implemented in a standalone package SpatialML. Both approaches abandon the spatial mixed model framework and try to account for the dependence structure in the data by incorporating additional spatial covariates. A downside to this is unnecessary escalation of

the problem to high-dimensional settings, being unable to parsimoniously encode structured spatial dependence via the GP covariance. This adversely affects the prediction performance of these methods compared to that of RF-GLS when there is a dominant covariate effect.

Additionally, these only focus on prediction and are unable to estimate the covariate effect (mean function) separately from the spatial effect, which can be of independent interest to the researcher. As far as time series analysis with RF is concerned, to the best of our knowledge, no dedicated `R` package is available at the moment (CRAN Task View: Time Series Analysis). The standard practice involves incorporating prior responses for a desired number of lags as covariates (Basak et al., 2019) and using block bootstrap for "bagging" to retain the local covariance structure. This data is then used with the classical RF for the purpose of forecasting. These approaches also suffer from similar problems as discussed before that are explicitly addressed in RF-GLS.

The RandomForestsGLS package

We provide a brief overview of the user functionality of the package. The package vignette delves deeper into this and demonstrates with example how the functions available in the package can be used for non-linear regression analysis of dependent data. Specific functions are discussed in detail in the code documentation of the package.

RF to RF-GLS: Accounting for correlation structure

In classical RF, which is an average of many regression trees, each node in a regression tree is split by optimizing the CART split criterion in Breiman et al. (1984). It can be rewritten in the following way.

$$v_n^{CART}((d,c)) = \frac{1}{n}\left(\|\mathbf{Y} - \mathbf{Z}^{(0)}\hat{\boldsymbol{\beta}}(\mathbf{Z}^{(0)})\|_2^2 - \|\mathbf{Y} - \mathbf{Z}\hat{\boldsymbol{\beta}}(\mathbf{Z})\|_2^2\right).$$

where, $\mathbf{Z}^{(0)}$ and $\mathbf{Z}$ are the membersip matrices for the leaf nodes of the tree before and after the potential node split. (d,c) denotes a potential cut (location of the split), with d and c being the cut direction (choice of the covariate) and cutoff point (value of the covariate) respectively, $\hat{\boldsymbol{\beta}}(\mathbf{Z})$ are the leaf node representatives given by OLS estimates corresponding to design matrix $\mathbf{Z}$ and can be written as:

$$\hat{\boldsymbol{\beta}}(\mathbf{Z}) = \left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1}\mathbf{Z}^\top\mathbf{y}$$

We observe that the split criterion is the difference of OLS loss functions before and after the cut with the design matrix of membership of the leaf nodes. We can incorporate the correlation structure of the data in the split criterion by replacing the OLS loss with GLS loss. The modified split criterion can be rewritten as:

$$\begin{aligned} v_{n,\mathbf{Q}}^{DART}((d,c)) = &\frac{1}{n}\Bigg[\left(\mathbf{Y} - \mathbf{Z}^{(0)}\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z}^{(0)})\right)^\top \mathbf{Q}\left(\mathbf{Y} - \mathbf{Z}^{(0)}\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z}^{(0)})\right) \\ &- \left(\mathbf{Y} - \mathbf{Z}\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z})\right)^\top \mathbf{Q}\left(\mathbf{Y} - \mathbf{Z}\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z})\right)\Bigg]. \end{aligned}$$

where, $\mathbf{Q}$ is the inverse of the working covariance matrix that models the spatial/serial dependence and $\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z})$ are the leaf node representatives given by the GLS estimates corresponding to design matrix $\mathbf{Z}$ and can be written as follows:

$$\hat{\boldsymbol{\beta}}_{GLS}(\mathbf{Z}) = \left(\mathbf{Z}^\top\mathbf{Q}\mathbf{Z}\right)^{-1}\mathbf{Z}^\top\mathbf{Q}\mathbf{y}.$$

Spatial Data

Model

We consider spatial point referenced data with the following mixed model:

$$y_i = m(\mathbf{x}_i) + w(\mathbf{s}_i) + \epsilon_i;$$

where, $y_i, \mathbf{x}_i$ respectively denotes the observed response and the covariate corresponding to the i^{th} observed location $\mathbf{s}_i$. $m(\mathbf{x}_i)$ denotes the covariate effect; spatial random effect, $w(\mathbf{s})$ accounts for spatial dependence beyond covariates modeled using a GP, and ϵ accounts for the independent and identically distributed random Gaussian noise.

Fitting

In the spatial mixture model setting, the package `RandomForestsGLS` allows for fitting $m(.)$ using RF-GLS using `RFGLS_estimate_spatial`. Spatial random effects are modeled using GP as is the practice. For model fitting, we use the computationally convenient Nearest Neighbor Gaussian Process (NNGP) (Datta et al., 2016). If the covariance parameters are unknown, they are automatically estimated from the specified covariance model and used in model fitting.

With coordinates `coords`, response `y` and the covariates `x`, we can fit RF-GLS for spatial data as:

```
est <- RFGLS_estimate_spatial(coords, y, x)
```

Prediction

Given a fitted model and the covariates `RFGLS_predict` predicts the covariate effects given a new covariate value. We also offer spatial prediction at new locations with `RFGLS_predict_spatial` which performs a non-linear kriging by combining the non-linear mean estimate from the covariate effects in `RFGLS_predict` and spatial kriging estimate from the BRISC package.

With new covariates `Xtest` and fitted RF-GLS model `est`, the covariate effect can be predicted using

```
RFGLS_predict <- RFGLS_predict(est, Xtest)
```

The spatial predictions at new locations `Coordstest`, can be obtained through

```
RFGLS_predict_spatial <- RFGLS_predict_spatial(est, Xtest, Coordstest)
```

Unknown model parameters

More often than not, the true covariance parameters of the data are not known apriori. Given a choice from a prespecified list of covariance functions (software presently allows for exponential, spherical, Matérn and Gaussian covariances), the software accommodates parameter estimation with the argument `param_estimate = TRUE`. For this, first we fit a classical RF ignoring the correlation structure. Next, we fit a zero mean Gaussian process with the covariance structure of choice on the residual to obtain the estimate of the model parameters through maximum likelihood estimation. This initial RF fitting and estimation of the spatial

parameters from the RF residuals is justified by theoretical results in Saha et al. (2021) proving that even under dependence, naive RF is consistent (although empirically suboptimal). The estimated model parameters are then used to fit RF-GLS. As demonstrated in Saha et al. (2021), this leads to significant improvement over classical RF both in terms of estimation and prediction accuracy. This procedure is again analogous to the linear model setup, where fitting a feasible GLS model often involves pre-estimating the covariance parameters from residuals from an OLS fit.

It has also been demonstrated (Section 4.4 in Saha et al. (2021)) that even under misspecification in covariance structure (i.e. when the true spatial effects are generated from a different covariance structure than the specified covariance for estimation or arbitrary smooth function), using RF-GLS with exponential covariance (the default choice of covariance function in the software) leads to noticeable improvement over simply using classical RF.

One additional consideration for model parameter estimation involves the computational complexity of the maximum likelihood estimation of Gaussian process. In its original form, this involves $O(n^3)$ computation and $O(n^2)$ storage space, which may be prohibitive for large n. This problem is thoroughly studied in spatial statistics literature (we refer the readers to Datta et al. (2016) for a detailed review). In the present scenario, we perform a fast, linear-time estimation of the model parameters with BRISC (Saha & Datta, 2018), which is directly implemented in the present software through the R package BRISC (https://CRAN.R-project.org/package=BRISC).

Autoregressive (AR) Time Series Data

Model

RF-GLS can also be used for function estimation in a time series setting under autoregressive (AR) errors. We consider time series data with errors from an AR(q) (autoregressive process of lag q) process as follows:

$$y_t = m(\mathbf{x}_t) + e_t;\ e_t = \sum_{i=1}^{q} \rho_i e_{t-i} + \eta_t$$

where, $y_i, \mathbf{x}_i$ denotes the response and the covariate corresponding to the t^{th} time point, e_t is an AR(q) process, η_t denotes the i.i.d. white noise and $(\rho_1, \cdots, \rho_q)$ are the model parameters that captures the dependence of e_t on $(e_{t-1}, \cdots, e_{t-q})$.

Fitting

In the AR time series scenario, the package `RandomForestsGLS` allows for fitting $m(.)$ using RF-GLS with `RFGLS_estimate_timeseries`. RF-GLS exploits the sparsity of the closed form precision matrix of the AR process for model fitting and prediction of mean function $m(.)$. If the AR model parameters (coefficients and order of autoregressive process) are unknown the code automatically estimates them from AR models with specified lags.

With response `y` and the covariates `x`, we can fit RF-GLS for temporal data as:

```
est <- RFGLS_estimate_timeseries(y, x)
```

Prediction of covariate effects

Prediction of covariate effects in AR process models are similar to that of spatial data and are performed with `RFGLS_predict` as in the case of spatial data.

Unknown model parameters

For time-series data, most often the model parameters for the autoregressive process (order of autoregression and the autocorrelation coefficients) are unknown. These are estimated using a strategy similar to the spatial case. With the option `param_estimate = TRUE`, the software estimates the model coefficients by fitting a zero mean autoregressive process of user defined order on the residuals from a naive RF fit. The autoregressive parameter fitting is done using `arima` from base R package `stats`.

Parallelization

For `RFGLS_estimate_spatial`, `RFGLS_estimate_timeseries`, `RFGLS_predict` and `RFGLS_predict_spatial` one can also take the advantage of parallelization, contingent upon the availability of multiple cores. One aspect of the parallelizatiion is through the multithreaded implementation of derivation of the NNGP components following `BRISC` package (which helps when we are dealing with a large dataset). We also run the regression trees on parallel, which can be beneficial when the number of trees is large. With very small dataset and small number of trees, communication overhead between the nodes for parallelization outweighs the benefits of the parallel computing. Hence it is recommended to parallelize only for moderately large dataset and/or large number of trees.

Package Features

- **Implementation**: The source code of the package are written in C/C++ for sake of optimizing execution time. The functions available to the user are wrappers around the source code, built with `R`'s foreign language interface. For the basic structure of the code, we make use of the open-source code of the regression trees in `R` based implementation of classical RF in `randomForest` package. As the split criterion in RF-GLS involves computationally intensive linear algebra operation in nested loops, we use `Fortran`'s Basic Linear Algebra Subprograms (BLAS) and Linear Algebra Package (LAPACK). This is achieved by storing all matrices in contiguous memory column-major format. We also offer multicore computation by building each regression tree independently.

- **NNGP approximation**: Node splitting in RF-GLS requires optimizing a cost function involving the Cholesky factor of the precision matrix. Use of the full dense precision matrix in spatial processes becomes taxing on typical personal computing resources both in terms of computational cost ($O(n^3)$) and storage cost ($O(n^2)$). In order to circumvent this problem, we use NNGP to replace the dense graph among spatial locations with a nearest neighbor graphical model. NNGP components can be combined to obtain a sparse cholesky factor, which closely approximates the decorrelation performance of the true cholesky. We implement a convenient nearest neighbor search following spNNGP (Finley et al., 2021) and efficient sparse matrix multiplication as in BRISC (Saha & Datta, 2018). The structure of the loops used in the process facilitates parallelization using `openMP` (Dagum & Menon, 1998) for this stage of calculation. In time series analysis, the sparsity in the precision matrix is inherently induced by AR covariance structure.

- **Scalable node splitting**: Another aspect of optimization of the proposed algorithm involves clever implementation of the cost function optimization. Provided candidate cut direction (d), the optimal cutoff point (c) is chosen by searching through the "gaps" in the corresponding covariate. Following the implementation of classical Regression Forest in randomForest, we start with a list of ordered covariate values corresponding to the prefixed candidate direction and assign them to one of the nodes initially. This

helps with searching through the "gaps" in the data, as searching the next "gap" is equivalent to switch of membership of the existing smallest member (w.r.t the covariate value in the prefixed direction) of the initially assigned node. In order to determine the cost function corresponding to each of the potential cutoff points, in each iteration, we serially switch the membership of the data points from the initial node. The process is graphically demonstrated in Figure 1.

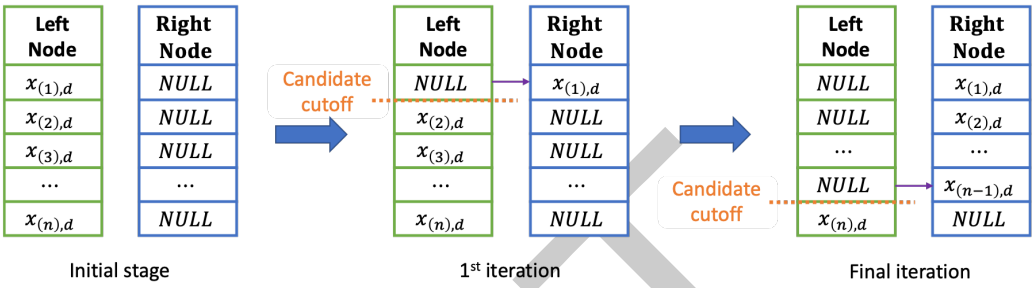

Figure 1: Serial update of membership and cutoff .

Since a serial update only affects one row in two columns of $\mathbf{Z}$ (corresponding to the newly formed nodes, an example of changes in $\mathbf{Z}$ corresponding to the changes in Figure 1 is demonstrated in Figure 2, here we note that the matrix components are unordered, hence the serial update will not follow the index ordering), the resulting correlation adjusted effective design matrix ($\mathbf{Q}^{1/2}\mathbf{Z}$) only experiences changes in the corresponding two columns and the rows corresonding to the points, which have this specific point in their nearest neighbor set. We efficiently implement this in the package which provides efficiency over brute force recomputation of the effective design matrix for each serial update. The process is graphically demonstrated in Figure 3.

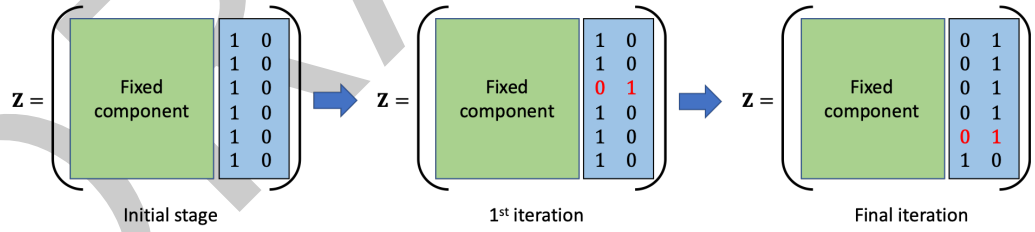

Figure 2: Serial update of Z .

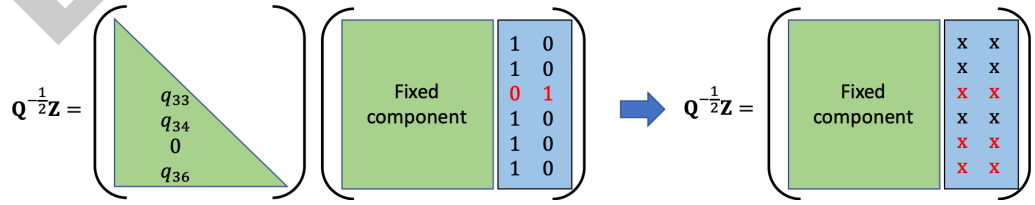

Figure 3: Changes in correlation adjusted design matrix .

Discussion

In this package, we have developed an efficient, parallel implementation of the RF-GLS method proposed in Saha et al. (2021). This accounts for the correlation in the data by incorporating

it in the node splitting criteria and the node representative update rule. The package accounts for spatial correlation, modeled with Matérn GP and serial autocorrelation. More often than not, the model parameters are unknown to the users, hence the package has inbuilt parameter estimation corresponding to both the covariance structures. Efficient implementation through C/C++ takes advantage of the NNGP approximation and scalable node splitting update rules which reduces the execution time. Present implementation of RF-GLS is slower than that of RF due to the additional computational complexity associated with the split criteria evaluation. In RF-GLS, evaluation of cost function, corresponding to a potential split has $O(t^3)$ computational complexity, where t is the number of leaf nodes at the present split. For very deep trees, this would lead to significant added computational burden over RF, where this step has $O(1)$ complexity. As a future extension of the package, implementation of covariate specific parallel optimization corresponding to each node splitting and improving overall computational complexity of the algorithm can be of independent research interest. We also plan to implement additional forms of time series dependency and perform thorough empirical validation, prior to making it available in the software.

Acknowledgements

AS and AD were supported by NSF award DMS-1915803. SB was supported by an NSF award DMS-1812128, and an NIH award R01GM135926.

References

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, *47*, 552–567. https://doi.org/10.1016/j.najef.2018.06.013

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press. https://doi.org/10.1201/9781315139470

Dagum, L., & Menon, R. (1998). OpenMP: An industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, *5*(1), 46–55. https://doi.org/10.1109/99.660313

Datta, A., Banerjee, S., Finley, A. O., & Gelfand, A. E. (2016). Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, *111*(514), 800–812. https://doi.org/10.1080/01621459.2015.1044091

Finley, A. O., Datta, A., & Banerjee, S. (2021). spNNGP: R package for nearest neighbor gaussian process models. *Journal of Statistical Software (Accepted Conditionally)*.

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., & Kalogirou, S. (2019). Geographical random forests: A spatial extension of the random forest algorithm to address spatial heterogeneity in remote sensing and population modelling. *Geocarto International*, 1–16. https://doi.org/10.1080/10106049.2019.1595177

Hengl, T., Nussbaum, M., Wright, M. N., Heuvelink, G. B., & Gräler, B. (2018). Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. *PeerJ*, *6*, e5518. https://doi.org/10.7717/peerj.5518

Saha, A., Basu, S., & Datta, A. (2021). Random forests for spatially dependent data. *Journal of the American Statistical Association*, *just-accepted*, 1–46. https://doi.org/10.1080/01621459.2021.1950003

Saha, A., & Datta, A. (2018). *BRISC: Fast inference for large spatial datasets using BRISC*. https://CRAN.R-project.org/package=BRISC