arXiv:1908.06936v1 [cs.DC] 23 Jul 2019

ExaGeoStatR: A Package for Large-Scale Geostatistics in R

Sameh Abdulah
King Abdullah Uni.
of Science and Tech.

Yuxiao Li
King Abdullah Uni.
of Science and Tech.

Jian Cao
King Abdullah Uni.
of Science and Tech.

Hatem Ltaief
King Abdullah Uni.
of Science and Tech.

David E. Keyes
King Abdullah Uni.
of Science and Tech.

Marc G. Genton
King Abdullah Uni.
of Science and Tech.

Ying Sun
King Abdullah Uni.
of Science and Tech.

Abstract

Parallel computing in Gaussian process calculation becomes a necessity for avoiding computational and memory restrictions associated with Geostatistics applications. The evaluation of the Gaussian log-likelihood function requires $O(n^2)$ storage and $O(n^3)$ operations where n is the number of geographical locations. In this paper, we present **ExaGeoStatR**, a package for large-scale Geostatistics in R that supports parallel computation of the maximum likelihood function on shared memory, GPU, and distributed systems. The parallelization depends on breaking down the numerical linear algebra operations into a set of tasks and rendering them for a task-based programming model. **ExaGeoStatR** supports several maximum likelihood computation variants such as exact, Diagonal Super Tile (DST), and Tile Low-Rank (TLR) approximation besides providing a tool to generate large-scale synthetic datasets which can be used to test and compare different approximations methods. The package can be used directly through the R environment without any C, CUDA, or MPI knowledge. Here, we demonstrate the **ExaGeoStatR** package by illustrating its implementation details, analyzing its performance on various parallel architectures, and assessing its accuracy using both synthetic datasets and a sea surface temperature dataset. The performance evaluation involves spatial datasets with up to 250K observations.

Keywords: Covariance matrix, Gaussian process, High performance computing, Large spatial dataset, Maximum likelihood estimation, Spatial prediction.

1. Introduction

Applications in spatial and spatio-temporal analytics deal with measurements regularly or irregularly located across a geographical region. Gaussian processes (GPs), or Gaussian random fields (GRFs), are the most valuable tools (Gelfand and Schliep 2016) in various applications by fitting the GP model to spatial datasets. GRFs also serve as building blocks for numerous non-Gaussian models in spatial statistics such as trans-Gaussian random fields, mixture GRF and skewed GRF. For example, Xu and Genton (2017) introduced the Tukey *g*-and-*h* random field which transforms the GRF by a flexible form of variable transformation; Andrews and Mallows (1974), West (1987), and Rue and Held (2005) mentioned the scale-mixture of Gaussian distributions and GRFs; and Allard and Naveau (2007) and Azzalini (2005) proposed many skewed GRFs and their variations. However, likelihood-based inference methods for GP models are computationally expensive for large spatial datasets. It is crucial to provide fast computational tools to fit a GP model to exascale spatial datasets that are often available in many real world applications.

Specifically, suppose Z is a stationary GRF with mean function $m(\cdot)$ and covariance function $C(\cdot, \cdot)$, and we observe data on a domain $D \subset \mathbb{R}^d$ at n locations, $\mathbf{s}_1, \ldots, \mathbf{s}_n$. Then, the random vector $\{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)\}^T$ follows a multivariate Gaussian distribution:

$$\forall \{\mathbf{s}_1, \ldots, \mathbf{s}_n\} \subset D, \quad \{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)\}^\mathrm{T} \sim \mathcal{N}_n(\boldsymbol{\mu}, \Sigma), \tag{1}$$

where $\boldsymbol{\mu} = \{m(\mathbf{s}_1), \ldots, m(\mathbf{s}_n)\}^\mathrm{T}$ and Σ are the mean vector and the covariance matrix of the n-dimensional multivariate normal distribution. Given $\boldsymbol{\mu}$ and Σ, the likelihood of observing $\mathbf{z} = \{z(\mathbf{s}_1), \ldots, z(\mathbf{s}_n)\}^\mathrm{T}$ at the n locations is

$$L(\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu})^\mathrm{T}\Sigma^{-1}(\mathbf{z} - \boldsymbol{\mu})\right\}. \tag{2}$$

The (i, j)-th element of Σ, $\Sigma_{ij} = C(\mathbf{s}_i, \mathbf{s}_j)$, where the covariance function $C(\mathbf{s}_i, \mathbf{s}_j)$ is assumed to have a parametric form with unknown vector of parameters $\boldsymbol{\theta}$. Various classes of valid covariance functions can be found in Cressie (2015). For simplicity, in this work, we assume the mean vector $\boldsymbol{\mu}$ to be zero to focus on estimating the covariance parameters. We choose the most popular isotropic Matérn covariance kernel, which is specified as,

$$\Sigma_{ij} = C(\|\mathbf{s}_i - \mathbf{s}_j\|) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\beta}\right)^\nu \mathcal{K}_\nu\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|}{\beta}\right), \tag{3}$$

where $\|\mathbf{s}_i - \mathbf{s}_j\|$ is the (Euclidean or great-circle) distance between $\mathbf{s}_i$ and $\mathbf{s}_j$, $\Gamma(\cdot)$ is the gamma function, $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of the second kind of order ν, and $\sigma^2, \beta > 0$, and $\nu > 0$ are the key parameters of the covariance function controlling the variance, spatial range, and smoothness, respectively. The Matérn covariance kernel is highly flexible and includes the exponential and Gaussian kernels as special cases. The variance, spatial range, and smoothness parameters, σ^2, β, and ν determine the properties of the GRF.

The typical inference for GRFs includes parameter estimation, stochastic simulation, and kriging (spatial prediction). Among these tasks, parameter estimation, or model fitting, is the primary and the most time-consuming one. Once the parameters are estimated, one can easily simulate multiple realizations of the GRF and obtain the prediction at unobserved locations. To obtain the maximum likelihood estimator (MLE), we need to optimize the

likelihood function in (2) over $\boldsymbol{\theta} = (\sigma^2, \beta, \nu)^{\mathrm{T}}$. However, the likelihood for a given $\boldsymbol{\theta}$ requires computing the covariance matrix Σ inside the likelihood function (2). It is performed by the Cholesky factorization of Σ, which requires $O(n^3)$ operations and $O(n^2)$ memory. This implies that the standard methods and traditional algorithms for GRFs are computationally infeasible for large datasets. On the other hand, technological advances in sensor networks along with the investments to data monitoring, collection, resource management provide massive open-access spatial datasets (Finley, Banerjee, and Gelfand 2015). Therefore, the unprecedented data availability and the challenging computational issues call for novel methods, algorithms, and software packages to deal with modern "Big Data" problems.

A broad literature focuses on developing efficient methodologies by approximating the covariance function in the GP model, so that the resulting covariance matrix is easier to compute. Sun, Li, and Genton (2012), Bradley, Cressie, and Shi (2016), and Liu, Ong, Shen, and Cai (2018) systematically reviewed these methods. Some popular approximation methods are covariance tapering (Furrer, Genton, and Nychka 2006; Kaufman, Schervish, and Nychka 2008), discrete process convolutions (Higdon 2002; Lemos and Sansó 2009), fixed rank kriging (Cressie and Johannesson 2008), lattice kriging (Nychka, Bandyopadhyay, Hammerling, Lindgren, and Sain 2015), and predictive processes (Banerjee, Gelfand, Finley, and Sang 2008; Finley, Sang, Banerjee, and Gelfand 2009). Meanwhile, some studies proposed to approximate the Gaussian likelihood function using conditional distributions (Vecchia 1988) or composite likelihoods (Varin, Reid, and Firth 2011; Eidsvik, Shaby, Reich, Wheeler, and Niemi 2014), and some seek for equivalent representation of GPs using spectral density (Fuentes 2007) and stochastic partial differential equations (Lindgren, Rue, and Lindström 2011).

A recent direction of this research aims at developing parallel algorithms (Paciorek, Lipshitz, Zhuo, Prabhat, Kaufman, and Thomas 2015; Katzfuss and Hammerling 2017; Datta, Banerjee, Finley, and Gelfand 2016; Guhaniyogi and Banerjee 2018) and using modern computational architectures, such as multicore systems, GPUs, and supercomputers, in order to avoid the over-smooth (Simpson, Lindgren, and Rue 2012; Stein 2014) problem in approximating GPs. Aggregating computing power through High-Performance Computing (HPC) becomes an important tool in scaling existing software in different fields to handle the exponential growth of datasets generated in these fields (Vetter 2013).

However, the literature is lacking a well-developed HPC software that can be used by the practitioners to support their applications with HPC capabilities. Although most of the studies provide reproducible source codes, they are difficult to be extended to new applications, especially when the algorithms require certain hardware setups. R is the most popular software in statistics, geostatistical analytics, and interactive exploration of data by far. As a higher-level language, however, R is relatively weak for high-performance computing compared to lower-level languages, such as C, C++, and Fortran. Scaling statistical software and bridging high-performance computing with the R language can be performed using two different strategies. One strategy is **pbdR** (Ostrouchov, Chen, Schmidt, and Patel 2012), Programming with Big Data in R, which transfers the HPC libraries to the R environment by providing a high-level R interface to a set of HPC libraries such as **MPI**, **ScaLAPACK**, **ZeroMQ**, to name a few. However, one drawback of this strategy is that the R developer should have enough background in HPC to be able to use the provided interfaces to scale his/her code. Another strategy which we adopt in this paper is to implement the statistical functions using an HPC-friendly language such as C. Then it is easier to directly wrap the C functions into R functions. In this case, these functions can directly be used inside the

R environment without the need of understanding the underlying HPC architectures or the development environment.

This paper presents **ExaGeoStatR**, a high-performance package in R for geostatistical applications, that depends on a unified C-based software called **ExaGeoStat** (Abdulah, Ltaief, Sun, Genton, and Keyes 2018b). **ExaGeoStat** is able to fit Gaussian process models and provide spatial predictions for geostatistics application in large-scale domains. **ExaGeoStat** provides both exact and approximate computations for large-scale spatial data. Besides the exact method, the software also supports two approximation methods, Diagonal Super Tile (DST) and Tile Low-Rank (TLR). This study aims at highlighting the capabilities of the **ExaGeoStatR** exact computations since it can be considered as a benchmark for the performance of other computation methods. Moreover, the evaluation of the DST and the TLR approximations has been already covered in Abdulah *et al.* (2018b) and Abdulah, Ltaief, Sun, Genton, and Keyes (2018a). The software also includes a synthetic dataset generator for generating large spatial datasets with the exact prespecified covariance function. Such large datasets can be used to perform broader scientific experiments related to large-scale computational geostatistics applications. Besides its ability to deal with different hardware architectures such as multicore systems, GPUs, and distributed systems, **ExaGeoStatR** utilizes the underlying hardware architectures to its full extent. Existing assessments on **ExaGeoStat** show the ability of the software to handle up to two million spatial locations on manycore systems (Abdulah *et al.* 2018a).

Existing R packages for fitting GRFs include **fields** (Nychka, Furrer, Paige, and Sain 2017), **geoR** (Ribeiro Jr and Diggle 2016), **spBayes** (Finley, Banerjee, and Carlin 2007; Finley *et al.* 2015), **RandomFields** (Schlather, Malinowski, Oesting, Boecker, Strokorb, Engelke, Martini, Ballani, Moreva, Auel, Menck, Gross, Ober, Ribeiro, Ripley, Singleton, Pfaff, and R Core Team 2019; Schlather, Malinowski, Menck, Oesting, and Strokorb 2015), **INLA** (Rue, Martino, and Chopin 2009; Martins, Simpson, Lindgren, and Rue 2013), **bigGP** (Paciorek *et al.* 2015). These packages feature different degrees of flexibility as well as computational capacity. The **spBayes** package fits GP models in the context of Bayesian or hierarchical modeling based on MCMC. The **RandomFields** package implements the Cholesky factorization method, the circulant embedding method (Dietrich and Newsam 1996), and an extended version of Matheron's turning bands method (Matheron 1973) for the maximum likelihood estimation of GRFs. The **INLA** package uses an integrated nested Laplace approximation to tackle additive models with a latent GRF, which outperforms the MCMC method. The **bigGP** package utilizes distributed memory systems through MPI (Gropp, Gropp, Lusk, and Skjellum 1999) to implement the estimation, prediction, and simulation of GRFs. The packages **fields** and **geoR** both estimate the GRF covariance structures designed for spatial statistics while **geoR** provides more flexibilities, such as estimating the mean structure and the variable transformation parameters. Among these popular packages, only the **bigGP** package, according to our knowledge, focuses on parallel computing, which is essential for solving problems in data-rich environments. Our package **ExaGeoStatR**, at the current stage, performs data generation, parameter estimation and spatial prediction for the univariate GRF with mean zero and a Matérn covariance structure, which is a fundamental model in spatial statistics. We feature breakthroughs in the optimization routine for the maximum likelihood estimation and the utilization of heterogeneous computational units. Specifically, we build on the optimization library NLOPT (Johnson 2014) and provide a unified API for multicore systems, GPUs, clusters, and supercomputers. The package also supports using the great circle distance in

constructing the covariance matrix. These parallelization features largely reduce the time-per-iteration in the maximum likelihood estimation compared with existing packages and make GRFs even with 10^6 locations estimable on hardware accessible to most institutions.

The remainder of this paper is organized as follows. Section 2 states the basics of **Exa-GeoStatR**, the user guide for the installation, and the package components for keen readers. Section 3 compares the estimation accuracy and time of **ExaGeoStatR** with two aforementioned exact-computation packages with simulated data and demonstrates the efficiency that can be gained when the **ExaGeoStatR** package utilizes powerful architectures including GPUs and distributed memory systems. Section 4 fits the Gaussian random field to a sea surface temperature dataset with more than ten thousand spatial locations per day and does kriging with the estimated parameter values. Section 5 concludes the contributions of the **ExaGeoStatR** package.

2. Tutorial and Implementation of the Software

2.1. Software Overview

ExaGeoStat[1] is a C-based high-performance software for geospatial statistics in climate and environment modeling (Abdulah *et al.* 2018b). This software provides a novel solution to deal with the scaling limitation impact of the MLE operation by maximizing the computational power of emerging hardware architectures. **ExaGeoStat** permits exploring the MLE computational limits using state-of-the-art high-performance dense linear algebra libraries by leveraging a single source code to run on various cutting-edge parallel architectures, e.g., Intel Xeon, Intel manycore Xeon Phi Knights Landing chip (KNL), NVIDIA GPU accelerators, and distributed-memory homogeneous systems.

ExaGeoStat software is not only developed to solve the maximum likelihood problem for a given set of data observed at n geographical locations in large scale but also to provide a prediction solution of unknown measurements at new locations. The software also allows for exact synthetic data generations with a given covariance function, which can be used to test and compare different approximation methods, for instance. To sum up, **ExaGeoStat** includes three main tools: large-scale synthetic data generator, the Gaussian maximum likelihood estimator, and the spatial predictor (kriging).

The maximum likelihood estimator includes three variant computation techniques for the covariance matrix: exact, Diagonal Super Tile (DST), and Tile Low-Rank (TLR). The exact solution provides dense algebraic computation by exploiting advances in solution algorithms and many-core computer architectures. Parallelization depends on dividing the given covariance matrix into a set of small tiles where each tile can be processed by a single processing unit. The DST approximation solution depends on annihilating off-diagonal tiles because their contributions, as well as their qualitative impact on the overall statistical problem, may be limited and depending on diagonal tiles which should have higher impact on the underlying model. The TLR approximation solution depends on exploiting the data sparsity of the dense covariance matrix by compressing the off-diagonal tiles up to a user-defined accuracy threshold (Abdulah *et al.* 2018a).

---

[1] https://github.com/ecrc/exageostat

(a) Exact computation.   (b) DST computation.   (c) TLR computation.
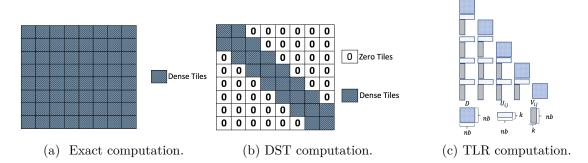
Figure 1: **ExaGeoStat** variant computation techniques: (a) Exact, (b) DST, (c) TLR.

Figure 1 shows different MLE computation techniques used in **ExaGeoStat**. In Figure 1a, exact computation is used where the generated covariance matrix is divided into a set of fully dense tiles. In Figure 1b, the DST approximation method uses the same execution strategy of the dense execution in addition to providing an approximation technique that depends on ignoring the off-diagonal tiles to speed up the overall computation time. In Figure 1c, the TLR approximation method is used where the k most significant singular values/vectors are captured for each off-diagonal tile to maintain the overall fidelity of the numerical model depending on the selected accuracy (i.e, application specific).

2.2. Software Infrastructure

ExaGeoStat internally relies on two state-of-the-art parallel linear algebra libraries to provide the basic linear algebra operations for the MLE operation. Exact and DST approximation computations relay on the **Chameleon** library, a high-performance numerical library, that provides high-performance solvers (cha 2019). The TLR approximation computation relies on the **HiCMA** library, a hierarchical linear algebra library on manycore architectures, that provides parallel approximation solvers (hic 2019). The **HiCMA** is associated with the **STARS-H** library, a high-performance $\mathcal{H}$-matrix generator library on large-scale systems, which provides test cases for the **HiCMA** library (sta 2019b). Both **Chameleon** and **HiCMA** libraries provide linear algebra operations through a set of sequential task-based algorithms.

To demonstrate the hardware portability of **ExaGeoStat**, it features a backend with the **StarPU** dynamic runtime system, which is preferred for its wide hardware architecture support (Intel manycore, NVIDIA GPU, and distributed-memory systems) (Augonnet, Thibault, Namyst, and Wacrenier 2011). **StarPU** proposes a kind of abstraction to improve both the productivity and creativity of the user. Since **Chameleon** and **HiCMA** provide sequential task-based through a sequential task flow (STF) programming model, **StarPU** is able to execute the set of given sequential tasks in parallel with given hints of the data dependencies (e.g., read, write, and read-write). The main advantage of using a runtime system that relies on task-based implementations such as **StarPU** is to become oblivious of the targeted hardware architecture. Multiple implementations of the same **StarPU** tasks are generated for: CPU, CUDA, OpenCL, OpenMP, MPI, to name a few. To achieve the highest performance, **StarPU** decides at runtime which implementation will achieve the highest performance. For the first execution, **StarPU** generates a set of cost models that determine the best hardware for optimal performance during the given tasks. This set of cost models may be saved for
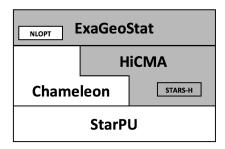
Figure 2: **ExaGeoStat** infrastructure.

future executions.

ExaGeoStat also relies on an open-source C/C++ nonlinear optimization toolbox, **NLopt** (Johnson 2014), to perform the MLE optimization operation. Among 20 global and local optimization algorithms supported by the **NLopt** library, we selected the Bound Optimization BY Quadratic Approximation (BOBYQA) algorithm to be our optimization algorithm because we are targeting a nonlinear problem with a global maximum point. BOBYQA is a numeric, global, derivative-free and bound-constrained optimization algorithm. It generates a new computed point on each iteration by solving a trust region subproblem subject to given constraints. In **ExaGeoStat** only upper and lower bound constraints are used. Though BOBYQA does not require the evaluation of the derivatives of the cost function, it employs an iteratively updated quadratic model of the objective, so there is an implicit assumption of smoothness.

ExaGeoStat relies on a set of software dependencies that expands the software portability capabilities. Figure 2 shows the **ExaGeoStat** infrastructure with three main layers, **ExaGeoStat**, which includes the upper-level functions of the software associated with the **NLopt** library for optimization purpose; the **Chameleon**/**HiCMA** libraries, which provide exact and approximate solvers for the linear algebra operations; and the **StarPU** runtime, which translates the software for execution on the appropriate underlying hardware. Table 1 summarizes the full list of software dependencies.

2.3. ExaGeoStatR

To facilitate the use of large-scale executions in R, we provide R-wrappers to our main **ExaGeoStat** computational functions through a separate package called **ExaGeoStatR**[2]. This R wrapper version should help in disseminating our software toward a large computational statistician community. To the best of our knowledge, most of existing R solutions for the MLE problem are sequential and restricted to limited data sizes.

Table 2 gives an overview of current **ExaGeoStatR** functions with a description of the main objective behind each function.

2.4. ExaGeoStatR Installation Tutorial

ExaGeoStatR is currently supported on both MacOS and Linux systems. Intel MKL library should be available on the target system before installing the **ExaGeoStatR** package.

---

[2]https://github.com/ecrc/exageostatR

Table 1: **ExaGeoStat** software dependencies.

Software	Description
hwloc	Portable Hardware Locality: provides a portable abstraction of the hierarchical topology of modern architecture.
NLopt	NonLinear optimization library: provides a common interface for several optimization algorithms implementations.
GSL	**GNU** Scientific Library: provides a set of numerical computing routines.
StarPU	A runtime system library for task-based programming model running on shared/distributed-system architectures as well as GPU-based systems.
Chameleon	A dense linear algebra software relying on sequential task-based algorithms and dynamic runtime systems.
HiCMA	Hierarchical Computations on Manycore Architectures: a low rank matrix computation library exploiting the data sparsity of the matrix operator.
STARS-H	Software for Testing Accuracy, Reliability, and Scalability of Hierarchical computations: a high performance low-rank matrix approximation library generating low-rank matrices on shared/distributed-memory systems.

Table 2: Overview of **ExaGeoStatR** functions.

Function Name	Description
`exageostat_init`	Initiate **ExaGeoStat** instance, defining the underlying hardware (i.e., number of cores and/or GPUs) and the tile size.
`simulate_data_exact`	Generate **Z** measurements vector on n unstructured random 2D locations.
`simulate_obs_exact`	Generate **Z** measurements vector on n given 2D locations.
`exact_mle`	Compute the MLE model parameters (exact computation).
`dst_mle`	Compute the MLE model parameters (DST approximation computation).
`tlr_mle`	Compute the MLE model parameters (TLR approximation computation).
`exageostat_finalize`	Finalize current active **ExaGeoStat** instance.

The following commands can be used to install **ExaGeoStatR** directly from GitHub:

```
>R library("devtools")
>R Sys.setenv(PKG_CONFIG_PATH = paste(Sys.getenv("PKG_CONFIG_PATH"),paste
        (.libPaths(),"exageostat/lib/pkgconfig",sep = '/',collapse = ':'),sep = ':'))
>R install_git(url="https://github.com/ecrc/exageostatR")
```

The *install_git* command can be edited to change the default configuration of the **ExaGeoStatR** package to support several installation modes:

To enable MPI support for distributed memory systems (i.e., MPI library should be available on the system).

```
>R install_git(url = "https://github.com/ecrc/exageostatR",
+       configure.args = c('--enable-mpi'))
```

To enable CUDA support for GPU systems (i.e., CUDA library should be available on the system).

```
>R install_git(url = "https://github.com/ecrc/exageostatR",
+       configure.args = c('--enable-cuda'))
```

If all **ExaGeoStatR** software dependencies have been installed on the system (i.e., install **ExaGeoStatR** package without dependencies).

```
>R install_git(url="https://github.com/ecrc/exageostatR",
+       configure.args = c('--no-build-deps'))
```

3. Simulation Studies

3.1. Performance Evaluation of ExaGeoStatR

In this section, we provide a set of examples with associated code for better understanding of the **ExaGeoStatR** package. The examples possess three goals: 1) provide step-by-step instructions of using **ExaGeoStatR** on multiple different tasks; 2) assess the performance and accuracy of the proposed exact computation compared to existing R packages; and 3) assess the performance of the **ExaGeoStatR** package using different hardware architectures.

The performance of **ExaGeoStatR** is evaluated on various systems: the experiments in examples 1 and 2 below are implemented on a Ubuntu 16.04.5 LTS workstation with a dual-socket 16-core Intel Sandy Bridge Intel Xeon E5-2650 without any GPU acceleration; example 3 is assessed on a dual-socket 18-core Intel Haswell Intel Xeon CPU E5-2698 v3 running at 2.30 GHz and equipped with 2 NVIDIA K80s (2 GPUs per board), and example 4 is tested on KAUST's Cray XC40 system, Shaheen II, with 6174 nodes, each node is dual-socket 16-core Intel Haswell processor running at 2.3 GHz and 128 GB of DDR4 memory. Two popular R packages, **fields** (Nychka *et al.* 2017) and **geoR** (Ribeiro Jr and Diggle 2016), are selected as our references for exact computations.

Since **ExaGeoStatR** works with multiple cores and different hardware architectures, users need to initialize their preferred settings using the `exageostat_init` function. When users want to change or terminate the current hardware allocation, the `exageostat_finalize` function is required:

```
>R library("exageostatr")
>R exageostat_init(hardware = list (ncores = 2, ngpus = 0, ts=320,
+       pgrid = 1, qgrid = 1))
>R exageostat_finalize()
```

The `hardware = list()` specifies the required hardware to execute the code. Here, `ncore` and `ngpu` are the numbers of CPU cores and GPU accelerators to deploy, `ts` denotes the tile size used for parallelized matrix operations, `pgrid` and `qgrid` are the cluster topology parameters in case of distributed memory system execution.

3.2. Performance Optimization Options

In general, the **ExaGeoStatR** performance on shared memory, GPUs, and distributed memory can be optimized by explicitly using **StarPU** optimization environment variables. For example, the `STARPU_SCHED` environment variable is used to select appropriate parallel tasks scheduling policies provided by **StarPU**, for example such as random, eager, and stealing. The user needs to try various schedulers to satisfy the best performance. Another example is `STARPU_LIMIT_MAX_SUBMITTED_TASKS` and `STARPU_LIMIT_MIN_SUBMITTED_TASKS` which control the number of submitted tasks and enable cache buffer reuse in main memory. A full list of **StarPU** environment variables can be found in sta (2019a).

3.3. Example 1: Data Generation

ExaGeoStatR offers two functions to generate Gaussian random fields with zero mean and Matérn covariance function shown in Equation (3). The `simulate_data_exact` function generates a GRF at a set of irregularly spaced random locations. Six parameters need to be given. The first three parameters are the initial model parameters, i.e., variance, spatial range, and smoothness, of the Matérn covariance function which are used to generate the simulated spatial dataset. `dmetric` is a boolean value and equals to zero in default. It can be set to zero for Euclidean distance, or one for great circle distance in case of sphere surface data (Veness 2010). `seed` is an integer value and equals to zero in default, used for pseudorandom number generations. The code below gives a simple example of generating a Gaussian random field at 1600 random locations using `simulate_data_exact` function.

```
>R exageostat_init(hardware = list(ncores = 2, ngpus = 0, ts = 320,
+        pgrid = 1, qgrid = 1))
>R data.exageo.irreg = simulate_data_exact(sigma_sq = 1, beta = 0.1,
+        nu = 0.5, dmetric = 0, n = 1600, seed = 0)
>R exageostat_finalize()
```

The results are stored as a list `data=list{x,y,z}`, where `x` and `y` are coordinates, and `z` is the simulated realizations. `x` and `y` here are generated from a uniform distribution on $[0,1]$. Therefore, the generated space is irregular on $[0,1] \times [0,1]$ with `simulate_data_exact` function. To generate data on regular grid, outside the range of $[0,1] \times [0,1]$, or on specific locations, one can use the `simulate_obs_exact` function by providing the coordinates `x` and `y`. The following code shows an example of generating a GRF on a gridded space on $[0,2] \times [0,2]$ with 1600 locations:

```
>R exageostat_init(hardware = list(ncores = 2, ngpus = 0, ts = 320,
+        pgrid = 1, qgrid = 1))
>R xy = expand.grid((1 : 40) / 20,(1 : 40) / 20)
>R x = xy[,1]
>R y = xy[,2]
```

```
>R data.exageo.reg = simulate_obs_exact(x = x, y = y, sigma_sq = 1,
+       beta = 0.1, nu = 0.5, dmetric = 0,  seed = 0)
>R exageostat_finalize()
```

For comparison reasons, we also show how the `sim.rf` function of **fields** and the `grf` function of **geoR** generate similar GRFs:

```
>R library(geoR)
>R sims = grf(n = 1600, grid = "reg", cov.pars = c(1, 0.1),
+       kappa = 0.5, RF = FALSE)
>R data.geoR.reg = list(x = sims$coords[, 1],
+       y = sims$coords[, 2], z = sims$data)

>R library(fields)
>R grid = list(x = (1 : 40) / 20,y = (1 : 40) / 20)
>R xy = expand.grid(x=(1 : 40) / 20, y=(1 : 40) / 20)
>R obj = matern.image.cov( grid = grid, theta = 0.1,
+       smoothness = 0.5, setup = TRUE)
>R sigma_sq = 1
>R sims.fields = sqrt(sigma_sq) * sim.rf(obj)
>R data.fields.reg = list(x = xy[, 1], y = xy[, 2],
+       z = c(sims.fields))
```

As can be seen, the three packages offer different types of flexibility in terms of data generation. However, when the goal is to generate a large GRF on an irregular grid with more than 20K locations, both the the `sim.rf` function and the `grf` function are not feasible. The `sim.rf` function simulates a stationary GRF only on a regular grid, and the `grf` function shows memory issues for the large size (`Error:vector memory exhausted`). On the other hand, the `simulate_data_exact` function can easily generate the GRF within one minute, which is implemented by the following code:

```
>R exageostat_init(hardware = list(ncores = 4, ngpus = 0, ts = 320,
+       pgrid = 1, qgrid = 1))
>R n = 25600
>R data = simulate_data_exact(sigma_sq, beta, nu, dmetric, n, seed)
>R exageostat_finalize()
```

Simulating data in large-scale requires enough memory and computation resources. Thus, we recommend the users to be consistent when generating large data with the available hardware resources. We also provide a set of synthetic and real large spatial data examples that can be downloaded from https://ecrc.github.io/exageostat/md_docs_examples.html for experimental needs.

3.4. Example 2: Performance on Shared Memory Systems for Moderate and Large Sample Size

To investigate the estimation of parameters based on the exact computation, we use the `exact_mle` function in **ExaGeoStatR**. On a shared memory system with a moderate sample

size, the number of cores `ncores` and tile size `ts` significantly affect the performance (see Figure 3). The following code shows the usage of the `exact_mle` function and returns execution time per iteration for one combination of n, `ncores` and `ts`:

```
>R exageostat_init(hardware = list (ncores = 2, ngpus = 0, ts = 160,
+       pgrid = 1, qgrid = 1))
>R data = simulate_data_exact(sigma_sq = 1, beta = 0.1,
+       nu = 0.5, dmetric = 0, n = 1600, seed = 0)
>R result = exact_mle(data, dmetric = 0, optimization =
+       list(clb = c(0.001, 0.001, 0.001), cub = c(5, 5, 5 ),
+               tol = 1e-4, max_iters = 20))
>R time= result$time_per_iter
>R exageostat_finalize()
```

In the `exact_mle` function, the first argument `data = list{x, y, z}` is a list that defines a set of locations in two dimensional coordinates `x,y`, and the measurement of the variable of interest `z`. `dmetric` is a distance parameter, the same as in the `simulate_data_exact` function. The `optimization` list specifies the optimization settings including the lower and upper bounds vectors, `clb` and `cub`, `tol` is the optimization tolerance and `max_iters` is the maximum number of iterations to terminate the optimization process. The optimization function uses the `clb` vector as the starting point of the optimization process.

The above example has been executed using three different sample sizes, 16 different numbers of cores, and four different tile sizes to assess the parallel execution performance of **ExaGeoStatR**. We visualize the results by using the `ggplot` function in **ggplot2** (Wickham 2016) as shown in Figure 3.

Figure 3 shows the computational time for the estimation process using a different number of cores up to 16 cores. The y-axis shows the total computation time per iteration in seconds while the x-axis represents the number of cores. The three sub-figures show the performance with different n values, 400, 900, and 1600. Different curves represent different tile size which
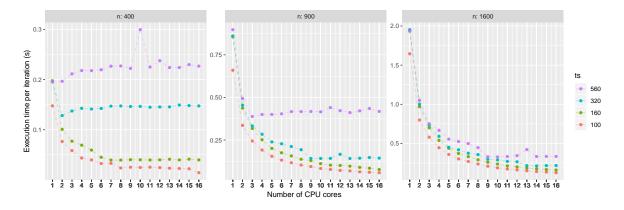

Figure 3: Parallel execution performance of **ExaGeoStatR** under different hardware settings. Each subfigure corresponds to a single sample size n and shows the execution time in seconds per iteration with regards to the number of cores up to 16. Curves with different colors provide the effect of tile size `ts`. (Red: `ts=100`. Green: `ts=160`. Blue: `ts=320`. Purple: `ts=560`)

impacts the performance on different hardware architectures. The figure shows that on our Intel Sandy Bridge machine the best-selected tile size is 100. We recommend trying different tile sizes to get the best performance out of the **ExaGeoStatR** package.

After specifying the hardware environment settings, we test the accuracy of the likelihood-based estimation of the **ExaGeoStatR** in comparison with **geoR** and **fields**. The counterparts to the `exact_mle` function are `likfit` in **geoR** and `MLEspatialProcess` in **fields**.

The simulated datasets from Example 1 are used as the input to assess the performance of parameter estimations. The synthetic datasets are generated on `n=1600` geometrical points in $[0, 1] \times [0, 1]$. We take the moderate sample size that costs **geoR** and **fields** approximately ten minutes, in order to obtain enough results with different scenarios and iterations. The mean structure is assumed to be constantly zero across the region. 100 replicates of samples are generated with different `seed` (`seed`$= 1, \ldots, 100$) to quantify the uncertainty. We estimate the parameter values for each sample and obtain 100 sets of estimates independently. A Matérn covariance kernel is selected to generate the covariance matrix with nine different scenarios. Specifically, the variance is always chosen to be one, `sigma_sq = 1.0`, the spatial range takes three different values representing high, medium, and low spatial correlation, `beta = c(0.3, 0.1, 0.03)`, and the smoothness also takes three values from rough to smooth, `nu = c(0.5, 1, 2)`. The simulated datasets with moderate sample size are generated by the `grf` function. We choose the `grf` function in **geoR** due to its flexibility in changing the parameter settings and in switching between regular and irregular grids.

We set the absolute tolerance to 10^{-5} and unset the maximum number of iterations (`max_iters = 0`) to avoid non-optimized results. Hence, each package can show its best performance to estimate the correct value of each parameter. The simulated GRFs are generated by the `grf` function and stored as a list called `data`:

```
>R library(geoR)
>R sigma_sq = 1
>R beta = 0.1 # choose one from c(0.3, 0.1,  0.03)
>R nu = 0.5 # choose one from c(0.5, 1, 2)
>R sims = grf(n = 1600, grid = "reg", cov.pars = c(sigma_sq, beta),
+        kappa = nu, RF = FALSE)
>R data = list(x = sims$coords[, 1], y = sims$coords[, 2], z = sims$data)
```

We have tried to keep the irrelevant factors as consistent as possible when comparing the **ExaGeoStatR** with **geoR** and **fields**. However, we identified some differences between the three packages that can hardly be reconciled. For example, **geoR** estimates the mean structure together with the covariance structure, and **fields** does not estimate the smoothness parameter, ν in our package. In addition, in terms of the optimization methods, both **geoR** and **fields** call the `optim` function in **stats** to maximize the likelihood function. The `optim` function include 6 methods such as `Nelder-Mead` and `BFGS`. However, **ExaGeoStatR** uses the BOBYQA algorithm, which is one of the optimization algorithms of the sequential **Nlopt** library in C/C++. The BOBYQA algorithm has the best performance in terms of MLE estimation, but it is not available in the `optim` function. Table 3 lists the differences between the three packages.

Multiple algorithms are offered by the `optim` function and further implemented by **geoR** and **fields**. However, many literatures point out that the `optim` function is not numerically stable for a large number of mathematical functions, especially when a re-parameterization

Table 3: Differences of the estimation function of **geoR**, **fields**, and **ExaGeoStatR**

Package	**geoR**	**fields**	**ExaGeoStatR**
Function name	`likfit`	`MLESpatialProcess`	`exact_mle`
Mean	estimated	estimated	fixed as zero
Variance	estimated	estimated	estimated
Spatial Range	estimated	estimated	estimated
Smoothness	estimated	fixed	estimated
Default optimization method	`Nelder-Mead`	`BFGS`[1]	`BOBYQA`[2]

[1.]BFGS: Broyden-Fletcher-Goldfarb-Shanno. [2.] BOBYQA: bound optimization by quadratic approximation

exists (Mullen *et al.* 2014; Nash, Varadhan *et al.* 2011; Nash *et al.* 2014). Based on the 100 simulated samples, we show that **ExaGeoStatR** not only provides faster computation, but also gives more accurate and robust estimations with regards to the initial value and grid type.

We first estimate the parameters by `exact_mle` in **ExaGeoStatR**. We use the number of cores to be 8 in order to reproduce the results on most of the machines. Users can specify their own settings and optimize the performance by referring to the results in Figure 3. The final results also report the time per iteration, total time, and the number of iterations for each optimization:

```
>R exageostat_init(hardware = list (ncores = 8, ngpus = 0, ts = 100,
+       pgrid = 1, qgrid = 1))
>R result = exact_mle(data, dmetric, optimization = list(clb =
+       c(0.001, 0.001, 0.001), cub = c(5, 5, 5 ), tol = 1e-4, max_iters = 0))
>R para_mat = c(result$sigma_sq, result$beta, result$nu)
>R time_mat = c(result$time_per_iter, result$total_time, result$no_iters)
>R exageostat_finalize()
```

Then we estimate the parameters under the same scenarios using the `likfit` function in **geoR** and the `MLEspatialProcess` function in **fields**. For **geoR** and **fields**, the chosen optimization options are `method = c("Nelder-Mead")`, `abstol = 1e-5`, and `maxit = 500`, where the maximum number of iterations is set as 500 which could never be reached. **fields** cannot optimize the smoothness ν, so we set it as the true value. **geoR** has to optimize the mean parameter, at least a constant, but it is treated to be independent of the covariance parameters as the mean value of data. In addition, to accelerate the optimization of **fields**, we minimize the irrelevant computation by setting `gridN = 1`:

```
>R result = MLESpatialProcess(x=cbind(data$x,data$y),y = data$z, cov.args =
+       list(Covariance = "Matern", smoothness = nu),
+       theta.start = 0.001, theta.range = c(0.001, 5), gridN = 1,
+       abstol = 1e-05, optim.args = list(method = c("Nelder-Mead"),
+       control = list(fnscale = -1, maxit = 500)))
>R para_mat = c(result$summary$rhoMLE,result$summary$theta,nu)
>R time_mat = c(result$timingTable[3,2]/dim(result$MLEJoint$lnLike.eval)[1],
+       result$timingTable[3,2],dim(result$MLEJoint$lnLike.eval)[1])
```

```
>R time = system.time( fit_obj = likfit(coords = cbind(data$x, data$y),
+        data = data$z, trend = "cte", ini.cov.pars = c(0.001, 0.001),
+        fix.nugget  = TRUE, nugget = 0, fix.kappa = FALSE, kappa = 0.001,
+        cov.model = "matern",  lik.method = "ML",
+        limits = pars.limits(sigmasq = c(0.001, 5),  phi = c(0.001, 5),
+        kappa = c(0.001, 5)), method = "Nelder-Mead",
+        control = list(abstol = 1e-5, maxit = 500)))[3]
>R time_mat[i,]=c(time / fit_obj$info.minimisation.function$counts[1],
+        time, fit_obj$info.minimisation.function$counts[1])
>R para_mat[i, ] = c(fit_obj$cov.pars, fit_obj$kappa )
```

The computational efficiency is compared based on the average execution time per iteration and the average number of iterations. As shown in Table 4, the running time per iterations of **ExaGeoStatR** is about 12 times and 7 times faster than **geoR** and **fields**, respectively. The running time per iteration is robust between different scenarios. Since **fields** does not estimate the smoothness parameter, it runs faster than **geoR**. Although **geoR** also estimates an extra constant mean parameter, it does not affect the computation much because the mean parameter is simply the mean of the measurements `z` and is estimated separately. Table 4 also shows the number of iterations to reach the tolerance. We can see that **ExaGeoStatR** requires more iterations but much less time to reach the accuracy.

Table 4: The average execution time per iteration and the average number of iterations to reach the tolerance based on 100 samples. Nine scenarios with three smoothness parameters, ν, and three spatial ranges, β, are assessed. The variance, σ^2, is set to be one.

The average execution time per iteration (seconds)									
Package	**geoR**			**fields**			**ExaGeoStatR**		
$\beta=$ $\nu=$	0.03	0.1	0.3	0.03	0.1	0.3	0.03	0.1	0.3
0.5	1.39	1.49	1.47	0.75	0.97	0.99	0.10	0.12	0.12
1	1.35	1.49	1.56	0.66	0.90	0.90	0.09	0.13	0.13
2	1.34	1.56	1.57	0.67	0.91	0.93	0.09	0.13	0.13
The average number of iterations to reach the tolerance									
$\beta=$ $\nu=$	0.03	0.1	0.3	0.03	0.1	0.3	0.03	0.1	0.3
0.5	160	157	135	73	72	70	231	204	237
1	193	33	23	75	75	80	318	320	275
2	216	25	20	100	70	85	427	436	332

Figure 4 shows the estimation accuracy between the three packages using boxplots. It is clear that **ExaGeoStatR** outperforms **geoR** and **fields**. Together with Table 4, we can see that **ExaGeoStatR** requires more iterations when ν increases since we set the initial values to be 0.001 for all scenarios. It implies that even under the circumstance of bad initial values, e.g. $\nu = 2$ and $\beta = 0.3$, **ExaGeoStatR** can reach the global maximum by taking more iterations. However, the estimation performance of both **geoR** and **fields** become worse when the initial values deviate from the truth. In particular, **geoR** reaches a local maximum only after about
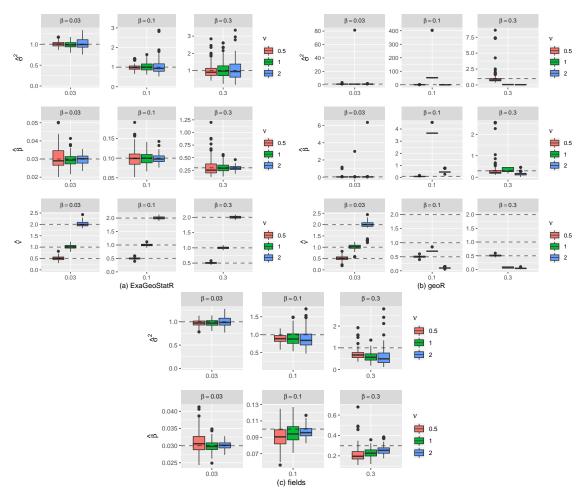
Figure 4: The estimation accuracy of (a) **ExaGeoStatR**, (b) **geoR**, and (c) **fields** with different set of parameter vectors. Each row shows the estimation of a parameter among variance, σ^2, spatial range, β, and smoothness, ν. The results related to **fields** only have two rows since the package does not estimate ν. Each column corresponds to one setting of spatial range, β, and the color of the boxplots identifies a type of smoothness, ν.

20 iterations for medium and large smoothness and spatial range, so that its estimated values are even not inside the range of **fields** and **ExaGeoStatR**.

The result is mainly due to the numerical optimization of ν, which involves the non-explicit Bessel function in the Matérn kernel. Therefore, **fields** has more robust estimation because it does not estimate the smoothness and we fix it as the truth, although **fields** calls the same `optim` function as **geoR**. However, eventhough **ExaGeoStatR** estimates the smoothness parameter, our package still outperforms **fields** in terms of β and σ^2, especially for medium and large spatial range.

Other optimization methods rather than "Neader-Mead" are also explored such as the default optimization option of **fields**, "BFGS". As a quasi-Newton method, "BFGS" is fast but not stable in many cases. Similar to the worst result of **geoR**, the optimization jumps out after only a few steps and reports a totally incorrect results even with a decent guess of initial values. Even worse, both **geoR** and **fields** report errors in computing the inverse of the covariance

matrix (`Error:error in solve.default(v$varcov, xmat):system is computationally singular`).

The experiment above shows that the `optim` function that both **geoR** and **fields** use is not stable with regards to the initial values, no matter what algorithm we choose. In addition, by simulating GRFs on an irregular grid (`grid = "irreg"`), few simulated GRFs can get an output without any error. For data on the irregular grid, locations can be dense so that the distance between certain points are too close to each other. Therefore, the columns of the covariance matrix corresponding to the dense locations can be numerically equal. For the data on a unit square, we find that **ExaGeoStatR** may only have the singularity problem when the closest distance is less than `1e-8`. On the contrary, the problem occurs when the smallest distance is close to `1e-4` for **fields** and **geoR**. As a result, although **ExaGeoStatR** is designed to provide a faster computation by making use of manycore systems, the optimization algorithm it is based upon gives a more accurate and robust estimation than any algorithm in the `optim` function.

We also investigate the computational time of the three packages as n increases. The hardware settings remain the same for **ExaGeoStatR** with 8 CPU cores. The max number of iterations are set to be 20 for all three packages to accelerate the estimation. The results report the total computational time. The tested number of locations ranges from 100 to 90,000. However, both **geoR** and **fields** take too much time when n is large. For example, the estimation with **geoR** and 22,500 locations requires more than 17 hours. Thus we stop the simulation for **geoR** and **fields** at the size of 22,500 and only show the execution time of **ExaGeoStatR** for larger n with 8 CPU cores. The results are shown in Figure 5. It can be seen that, when $n = 22,500$, **ExaGeoStatR** is 33 times faster than **fields** and 92 times faster than **geoR**.
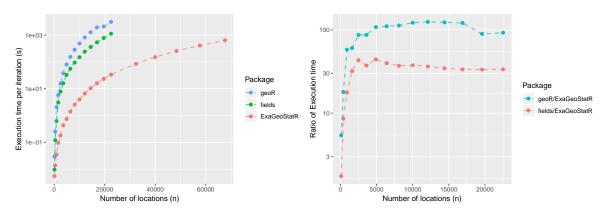

Figure 5: The execution time per iterations as n increases for **geoR**, **fields**, and **ExaGeoStatR** with 8 cores. The covariance parameters are set to be $(\sigma^2, \beta, \nu) = (1, 0.1, 0.5)$. Each curve on the left panel shows the exact running time per iteration of one package, and each curve on the right panel gives the ratio of execution time compared to **ExaGeoStatR**. The y-axis is shown in $\log_{10}$ scale.

3.5. Example 3: Extreme Computing on GPU Systems

Since the main advantage of **ExaGeoStatR** is the multi-architecture compatibility, here we provide another example of using **ExaGeoStatR** on GPUs systems. We choose the number of locations ranging from 1600 to approximately 100K. The R code below shows an example of

how to use 26 CPU cores and 1 GPU core with $n = 25,600$.

```
>R exageostat_init(hardware = list (ncores = 26, ngpus = 1,
+        ts=960, pgrid=1, qgrid=1))
>R n = 25600
>R data = simulate_data_exact(sigma_sq = 1,
+        beta = 0.1, nu = 0.5, dmetric = 0, n, seed = 0)
>R result_cpu = exact_mle(data, dmetric, max_iters = 20)
>R time_cpu = result_cpu$time_per_iter
>R exageostat_finalize()
```

From the above code, one can see that **ExaGeoStatR** has a user friendly interface to abstract the underlying hardware architecture to the user. The user needs only to specify the number of cores and GPUs required for one's execution.

Figure 6 reports the performance with different numbers of GPU accelerators: 1, 2, and 4. The figure also shows the curve using the maximum number of cores (28-core) on the machine without any GPU support. The figure demonstrates how using GPUs speeds up the execution time compared to the CPU-based running. Moreover, the figure shows the scalability using different numbers of GPUs.

Figure 6: Execution performance of **ExaGeoStatR** under different CPU and GPU combinations. The covariance parameters are set to be $(\sigma^2, \beta, \nu) = (1, 0.1, 0.5)$. Each line corresponds to the execution time per iteration with regards to different sample size n.

3.6. Example 4: Extreme Computing on Distributed Memory Systems

In this subsection, we give an example of using **ExaGeoStatR** on distributed memory systems (i.e., Shaheen II Cray XC40). As for shared memory and GPU systems, the **ExaGeoStatR** package abstracts the underlying hardware to a set of parameters. With distributed systems, the user needs to define four main parameters: `pgrid` and `qgrid` which represent a set of nodes arranged in a `pgrid` × `qgrid` rectangular array of nodes (i.e, two-dimensional block-cyclic distribution), `ncores` which represents the number of physical cores in each node, and

`ts` which represents the optimized tile size. Another example of the usage of **ExaGeoStatR** on a distributed memory system with 31 CPU cores, 4×4 rectangular array of nodes, `ts=960` and, `n=250,000` is shown below:

```
>R exageostat_init(hardware = list (ncores = 31, ts=960,
+        pgrid=4, qgrid=4))
>R n = 250000
>R data = simulate_data_exact(sigma_sq = 1,
+        beta = 0.1, nu = 0.5, dmetric = 1, n, seed = 0)
>R result_cpu = exact_mle(data, dmetric, max_iters = 20)
>R time_cpu = result_cpu$time_per_iter
>R exageostat_finalize()
```

Figure 7 shows the performance results of running different problem sizes on Shaheen II Cray XC40 using different numbers of nodes. The distribution of the nodes are 2×2, 4×4, 8×4, 8×8, and 16×16. The figure shows strong scalability of **ExaGeoStatR** with different numbers of nodes up to 64 nodes. The reported performance is the time per iteration averaged over 20 iterations with setting `STARPU_SCHED=eager` and `STARPU_LIMIT_MAX_SUBMITTED_TASKS=10000` and `STARPU_LIMIT_MIN_SUBMITTED_TASKS=9000`.

Figure 7: Performance of **ExaGeoStatR** using different numbers of nodes. The time per iteration is averaged over 20 iterations. The realization is generated from a zero-mean GRF under the Matérn covariance structure with the parameters $(\sigma^2, \beta, \nu) = (1, 0.1, 0.5)$.

4. Application to Sea Surface Temperature Data

West-blowing trade winds in the Indian Ocean push warm surface waters against the eastern coast of Africa. These waters move south along the coastline, eventually spilling out along the boundary of the Indian and Atlantic Oceans. This jet of warm water, known as the Agulhas Current, collides with the cold, west to east flowing Antarctic Circumpolar Current, producing a dynamic series of meanders and eddies as the two waters mix. The result makes for an interesting target for spatial analysis.

This application study provides an example where the MLE is computed in high dimensions and **ExaGeoStatR** facilitates the procedure on many-core systems. We use the sea surface temperature collected by satellite for the Agulhas and surrounding areas off the shore of South Africa. The data covers 331 days, from January 1 to November 26, 2004. The region is abstracted into a 72×240 regular grid, with the grid lines denoting the latitudes and longitudes and the spatial resolution is approximately 25 kilometers, though exact values depend on latitude. Although **fields** and **geoR** do not have input dimension limits, the computation with **ExaGeoStatR** has a distinct advantage on parallel architectures, hence more suitable for MLE with more optimization iterations to reach convergence.

Our analysis considers only the spatial structure in the spatio-temporal dataset and hence, assumes independence between the parameters on different days. Before introducing our model, we first present some exploratory data analysis. In Figure 9a, we use the `image.plot` function from the **fields** package to plot the heatmap of the sea surface temperature on selected four days that are also used for showing the kriging results later. Numerous gaps are present in the data, corresponding to three main causes: 1) land: specifically South Africa and Lesotho, visible in the left-center of the top of the plot, as well as two small islands towards the southern boundary; 2) clipping: the large wedge-shaped voids cutting N-S across the picture resulting from the satellite's orbital path; and 3) cloud cover: all or most of the remaining swirls and dots present in the image. Various forms of kriging can be used to attempt to fill those gaps caused by orbital clipping and cloud cover. Of course it does not make sense to estimate sea surface temperatures for gaps caused by the presence of land. A pronounced temperature gradient is visible from highs of over 25° C in the north of the study area to a low of 3.5° C towards the southern boundary. This is not only indicative of spatial correlation in the dataset, but it also shows that the data are not stationary, as the mean temperature must vary strongly with latitude.

We plot the mean and standard deviation along each latitude on the four days in Figure 8. The longitudinal mean and standard deviation (not shown) are relatively stable although there are spikes and troughs due to the missing data. Since the locations are sufficient for a regression with only three parameters, we also include the longitude as a regression variable and assume the following linear model with Gaussian noise for the sea surface temperature,
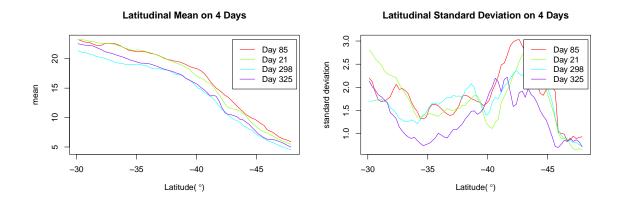

Figure 8: Exploratory data analysis on Days 85, 21, 298, and 325. The mean and standard deviation are computed with missing values removed.

$T(\lambda, \alpha)$:

$$T(\lambda, \alpha) = \mu(\lambda, \alpha) + \epsilon(\lambda, \alpha; \sigma^2, \beta, \nu),$$
$$\mu(\lambda, \alpha) = c + a \cdot \lambda + b \cdot \alpha,$$

where λ denotes longitude, α denotes latitude, and $\epsilon(\lambda, \alpha)$ is a GRF with zero mean and a Matérn covariance structure parameterized the same way as before. Here, a, b, and c are the linear coefficients for the mean structure, which we compute prior to the covariance structure based on the least square estimation. Maximum likelihood estimation is only applied to σ^2, β, and ν at the second stage because non-convex optimization for six parameters requires a larger sample size and significantly more iterations. The original data have different proportions of missing values, varying from Day 1 to Day 331. We ignore those days whose missing value proportion exceeds 50% so that the number of predictions is not more than the original number of observations.

The model fitting is done with the `exact_mle` function from the **ExaGeoStatR** package, which maximizes the exact likelihood. Specifically, the function call is:

```
>R x = x[!is.na(z)]
>R y = y[!is.na(z)]
>R z = z[!is.na(z)]
>R mydf = data.frame(x, y, z)
>R mymdl = lm(formula = z ~ x + y , data = mydf)
>R z = as.numeric(mymdl$residuals)
>R mytime = system.time(theta_out = exact_mle(data, dmetric,
+        optimization = list(clb = c(0.001, 0.001, 0.001),
+        cub = c(5, 5, 5 ), tol = 1e-4, max_iters = 20))[3]
```

Referring to Section 3, `n` is the number of spatial locations, `ncore` and `ngpu` are the numbers of CPU cores and GPU accelerators to deploy, `ts` denotes the tile size used for parallelized matrix operations, `pgrid` and `qgrid` are the cluster topology parameters, `x` and `y` store either the cartesian coordinates or the spherical coordinates of the geometry, `z` is one realization of the spatial variables of dimension `n`, `clb` and `cub` define the search range for the three parameters, `dmetric` is a boolean indicating whether the Euclidean distance or the great circle distance is used for the covariance matrix, `tol` and `niter` specify the stopping criteria which supports both a tolerance level for reckoning convergence and the maximum number of iterations. For this application study, 16 Intel Sandy Bridge Xeon E5-2650 processors are used without any GPU acceleration.

The tile size is initialized at 160 and the dimensions of the grid are both 1 for simplicity. In order to compare with the **geoR** and **fields** packages, we set `niter` to 20 and measure the time cost of fitting the GRF to the data on Day 1, which has over 8,800 valid (not NA) locations. The following is the code calling the `likfit` and `MLESpatialProcess` functions while the function call for `exact_mle` is already shown above:

```
>R time = system.time(fit_obj = MLESpatialProcess(cbind(x, y),
+        z,cov.args = list(Covariance = "Matern", smoothness = 0.8),
+        verbose = T, theta.start = 0.1, theta.range = c(0.1, 5),
+        optim.args = list(method = "Nelder-Mead",
```

```
+          control = list(maxit = 20, tol = 1e-4))))[3]
>R data_obj = as.geodata(cbind(x, y, z))
>R time = system.time(fit_obj = likfit(geodata = data_obj, trend = "cte",
+          ini.cov.pars = c(0.1, 0.1), fix.nugget = TRUE, nugget = 0,
+          fix.kappa = FALSE, kappa = 0.1, cov.model = "matern",
+          lik.method = "ML", limits = pars.limits(sigmasq = c(0.01, 20),
+          phi = c(0.01, 20), kappa = c(0.01, 5)), print.pars = TRUE,
+          method = "Nelder-Mead", control = list(maxit = 20,abstol = 1e-4)))[3]
```

The `exact_mle` function was executed with 15 CPUs and took 147 seconds, the `likfit` function from the **geoR** package cost 2,286 seconds, and the `MLESpatialProcess` function from the **fields** package needed 4,049 seconds. It usually requires more than 100 iterations to reach convergence, which renders the **geoR** and **fields** packages very difficult to fit high-dimensional GRFs, whereas the **ExaGeoStatR** package utilizes parallel architectures and reduces the time cost by more than one order of magnitude. Hence, **ExaGeoStatR** allows to fill many spatial images quickly.

We select $(0.01, 20.0)$ as the searching range for σ^2 and β and $(0.01, 5.00)$ for ν to guarantee the results not landing on boundary values. The initial values for all three parameters are the corresponding lower bounds of the searching ranges by default and the optimization continues without any limit on the number of iterations until convergence is reached within a tolerance level of 10^{-4}. There are 174 days whose missing value percentages are below 50% and Table 5 summarizes the independently estimated parameters for these days. Here, ν has the most

Table 5: Summary statistics for the estimated parameters across 174 days of sea surface temperature.

	Min	25% Q	Median	Mean	75% Q	Max
σ^2	3.41	5.78	6.44	6.33	6.76	14.40
β	1.99	2.76	3.02	3.03	3.27	4.60
ν	0.81	0.89	0.91	0.91	0.93	1.00

consistent estimations among the three parameters while σ^2 and β have similar variability. Based on the estimated parameters, we predict the sea surface temperature at locations where the data are missing with kriging, which computes the conditional expectation using a global neighborhood. The kriging is done with the `krige.conv` function from the **geoR** package as indicated below:

```
>R data_obj = as.geodata(cbind(x_known,y_known,z_known))
>R krig_ctl = krige.control(cov.model = "matern",
+          cov.pars = c(theta_out[r, 1],theta_out[r, 2]),
+          kappa = theta_out[r, 3])
>R krig_obj = krige.conv(geodata = data_obj,
+          locations = cbind(x_unknown, y_unknown),
+          krige = krig_ctl)
```

In Figure 9 we show the original and the predicted sea surface temperature for the four days corresponding to the 99%, 66%, 33%, and 1% quantiles of the estimated ν to visualize the
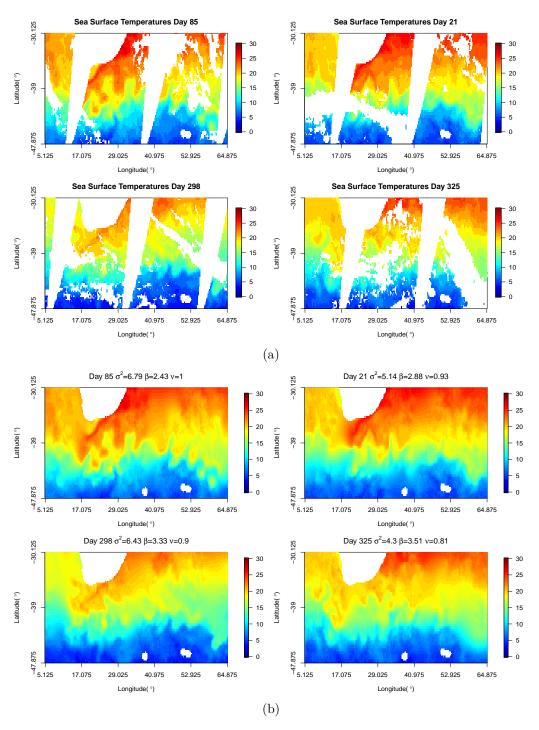
(a)

(b)

Figure 9: Panels (a) are the original sea surface temperatures in Celsius (°C) where the locations with NA values are colored in white. Panels (b) are the predicted sea surface temperatures in Celsius (°C) based on the linear mean structure and the kriging results where the land area is not predicted and colored in white. Parameter estimates are provided for each panel.

smoothness change. Day 85 seemingly has more details than Day 325 although the main factor governing the temperature change is the mean structure.

5. Discussion

In this paper, we presented the **ExaGeoStatR** package for large scale Geostatistics. The package provides a parallel computation for Gaussian maximum likelihood function evaluation using shared memory, GPUs, and distributed systems. Large-scale Gaussian calculations in R become possible with **ExaGeoStatR** by mitigating its memory space and computing restrictions.

We also analyzed and assessed the performance of exact computation variants of **ExaGeoStatR** against existing well-known R packages, such as **geoR** and **fields**. The evaluation shows a large difference in **ExaGeoStatR** performance compared to the other two packages. The accuracy evaluation also shows that **ExaGeoStatR** performs very well in different synthetic datasets compared to **geoR** and **fields**. We focused on exact computations to show the advantage of **ExaGeoStatR** over the aforementioned R packages and its ability to run on different existing hardware architectures. However, **ExaGeoStatR** is not limited by exact computations. It also includes two approximation methods, Diagonal Super Tile (DST) and Tile-Low Rank (TLR). The evaluation of these two methods have been already covered in Abdulah *et al.* (2018b,a). Moreover, the package is designed to be extensible to other approximation methods of Gaussian process calculations in the future.

We aimed from the beginning to abstract the parallel execution functions to the R developer. The developer needs only to specify some parameters to define the underlying hardware architecture and the package will take care of optimizing the execution on the target hardware. In this way, we increased the portability of our software and made it more suitable for R community developers.

The current version of **ExaGeoStatR** only supports a zero mean and an isotropic Matérn covariance function to provide a robust and efficient estimation of covariance. However, the package can also be helpful in many other problems. First, when the mean function is not zero, the simplest way is to estimate the mean and the covariance function independently as we did in the application. Theoretically, this independent maximum likelihood estimation will result in a biased random effect and can be improved by the restricted maximum likelihood (REML) techniques. However, as **fields** suggests, using REML typically does not make much difference. Second, we can make prediction at unknown locations (kriging) with uncertainties once the covariance parameters are fitted. The prediction is calculated by the conditional distribution of the multivariate Gaussian. Third, even when spatial nonstationarity is observed, we can still apply the **ExaGeoStatR** by assuming local stationarity. This idea is implemented in **convoSPAT** (Risser and Calder 2017). Once we obtain the estimated parameters locally, we can reconstruct the nonstationary covariance function. Finally, **ExaGeoStatR** is also useful for space-time and multivariate GRFs, where the covariance function we use should be replaced by the spatio-temporal covariance function and the cross-covariance function, respectively. As our future work, **ExaGeoStatR** will provide the necessary built-in functions to support the aforementioned extensions for more complex applications.

6. Acknowledgments

References

(2019). "**Chameleon**, a dense linear algebra software for heterogeneous architectures." `http://project.inria.fr/`. [Online; accessed May 2019].

(2019a). "**StarPU**: User Guide." `http://starpu.gforge.inria.fr/doc/starpu.pdf`. [Online; accessed May 2019].

(2019b). "**STARS-H**, a high performance parallel software for testing accuracy, reliability and scalability of hierarchical computations." `https://github.com/ecrc/stars-h`. [Online; accessed May 2019].

(2019). "The Hierarchical Computations on Manycore Architectures (**HiCMA**) library." `https://github.com/ecrc/hicma`. [Online; accessed May 2019].

Abdulah S, Ltaief H, Sun Y, Genton MG, Keyes DE (2018a). "Parallel Approximation of the Maximum Likelihood Estimation for the Prediction of Large-Scale Geostatistics Simulations." In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 98–108. IEEE.

Abdulah S, Ltaief H, Sun Y, Genton MG, Keyes DE (2018b). "**ExaGeoStat**: A High Performance Unified Software for Geostatistics on Manycore Systems." *IEEE Transactions on Parallel and Distributed Systems*, **29**(12), 2771–2784.

Allard D, Naveau P (2007). "A New Spatial Skew-normal Random Field Model." *Communications in Statistics—Theory and Methods*, **36**(9), 1821–1834.

Andrews DF, Mallows CL (1974). "Scale Mixtures of Normal Distributions." *Journal of the Royal Statistical Society: Series B (Methodological)*, **36**(1), 99–102.

Augonnet C, Thibault S, Namyst R, Wacrenier PA (2011). "**StarPU**: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures." *Concurrency and Computation: Practice and Experience*, **23**(2), 187–198.

Azzalini A (2005). "The Skew-normal Distribution and Related Multivariate Families." *Scandinavian Journal of Statistics*, **32**(2), 159–188.

Banerjee S, Gelfand AE, Finley AO, Sang H (2008). "Gaussian Predictive Process Models for Large Spatial Data Sets." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **70**(4), 825–848.

Bradley JR, Cressie N, Shi T (2016). "A Comparison of Spatial Predictors when Datasets could be Very Large." *Statistics Surveys*, **10**, 100–131.

Cressie N (2015). *Statistics for Spatial Data.* John Wiley & Sons.

Cressie N, Johannesson G (2008). "Fixed Rank Kriging for Very Large Spatial Data Sets." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **70**(1), 209–226.

Datta A, Banerjee S, Finley AO, Gelfand AE (2016). "Hierarchical Nearest-neighbor Gaussian Process Models for Large Geostatistical Datasets." *Journal of the American Statistical Association*, **111**(514), 800–812.

Dietrich C, Newsam G (1996). "A Fast and Exact Method for Multidimensional Gaussian Stochastic Simulations: Extension to Realizations Conditioned on Direct and Indirect Measurements." *Water Resources Research*, **32**(6), 1643–1652.

Eidsvik J, Shaby BA, Reich BJ, Wheeler M, Niemi J (2014). "Estimation and Prediction in Spatial Models with Block Composite Likelihoods." *Journal of Computational and Graphical Statistics*, **23**(2), 295–315.

Finley AO, Banerjee S, Carlin BP (2007). "**spBayes**: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models." *Journal of Statistical Software*, **19**(4), 1–24. URL http://www.jstatsoft.org/v19/i04/.

Finley AO, Banerjee S, Gelfand AE (2015). "**spBayes** for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models." *Journal of Statistical Software*, **63**(13), 1–28. URL http://www.jstatsoft.org/v63/i13/.

Finley AO, Sang H, Banerjee S, Gelfand AE (2009). "Improving the Performance of Predictive Process Modeling for Large Datasets." *Computational Statistics and Data Analysis*, **53**(8), 2873–2884.

Fuentes M (2007). "Approximate Likelihood for Large Irregularly Spaced Spatial Data." *Journal of the American Statistical Association*, **102**(477), 321–331.

Furrer R, Genton MG, Nychka D (2006). "Covariance Tapering for Interpolation of Large Spatial Datasets." *Journal of Computational and Graphical Statistics*, **15**(3), 502–523.

Gelfand AE, Schliep EM (2016). "Spatial Statistics and Gaussian Processes: A Beautiful Marriage." *Spatial Statistics*, **18**, 86–104.

Gropp WD, Gropp W, Lusk E, Skjellum A (1999). *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, volume 1. MIT press.

Guhaniyogi R, Banerjee S (2018). "Meta-kriging: Scalable Bayesian Modeling and Inference for Massive Spatial Datasets." *Technometrics*, **60**(4), 430–444.

Higdon D (2002). "Space and Space-time Modeling Using Process Convolutions." In *Quantitative Methods for Current Environmental Issues*, pp. 37–56. Springer.

Johnson S (2014). "The **NLopt** Nonlinear-optimization Package [Software]."

Katzfuss M, Hammerling D (2017). "Parallel Inference for Massive Distributed Spatial Data using Low-rank Models." *Statistics and Computing*, **27**(2), 363–375.

Kaufman CG, Schervish MJ, Nychka DW (2008). "Covariance Tapering for Likelihood-based Estimation in Large Spatial Data Sets." *Journal of the American Statistical Association*, **103**(484), 1545–1555.

Lemos RT, Sansó B (2009). "A Spatio-temporal Model for Mean, Anomaly, and Trend Fields of North Atlantic Sea Surface Temperature." *Journal of the American Statistical Association*, **104**(485), 5–18.

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: Series B (Statistical Methodology)*, **73**(4), 423–498.

Liu H, Ong YS, Shen X, Cai J (2018). "When Gaussian Process meets Big Data: A Review of Scalable GPs." *arXiv preprint arXiv:1807.01065*.

Martins TG, Simpson D, Lindgren F, Rue H (2013). "Bayesian Computing with **INLA**: New Features." *Computational Statistics & Data Analysis*, **67**, 68–83.

Matheron G (1973). "The Intrinsic Random Functions and Their Applications." *Advances in Applied Probability*, **5**(3), 439–468.

Mullen KM, *et al.* (2014). "Continuous Global Optimization in R." *Journal of Statistical Software*, **60**(6), 1–45.

Nash JC, Varadhan R, *et al.* (2011). "Unifying Optimization Algorithms to Aid Software System Users: **optimx** for R." *Journal of Statistical Software*, **43**(9), 1–14.

Nash JC, *et al.* (2014). "On Best Practice Optimization Methods in R." *Journal of Statistical Software*, **60**(2), 1–14.

Nychka D, Bandyopadhyay S, Hammerling D, Lindgren F, Sain S (2015). "A Multiresolution Gaussian Process Model for the Analysis of Large Spatial Datasets." *Journal of Computational and Graphical Statistics*, **24**(2), 579–599.

Nychka D, Furrer R, Paige J, Sain S (2017). "**fields**: Tools for Spatial Data." doi:10.5065/D6W957CT. R package version 9.7, URL github.com/NCAR/Fields.

Ostrouchov G, Chen WC, Schmidt D, Patel P (2012). "Programming with Big Data in R." URL http://r-pbd.org/.

Paciorek CJ, Lipshitz B, Zhuo W, Prabhat, Kaufman CG, Thomas RC (2015). "Parallelizing Gaussian Process Calculations in R." *Journal of Statistical Software*, **63**(10), 1–23. URL http://www.jstatsoft.org/v63/i10/.

Ribeiro Jr PJ, Diggle PJ (2016). **geoR**: *Analysis of Geostatistical Data.* R package version 1.7-5.2, URL https://CRAN.R-project.org/package=geoR.

Risser MD, Calder CA (2017). "Local Likelihood Estimation for Covariance Functions with Spatially-Varying Parameters: The convoSPAT Package for R." *Journal of Statistical Software*, **81**(14), 1–32. doi:10.18637/jss.v081.i14.

Rue H, Held L (2005). *Gaussian Markov Random Fields: Theory and Applications.* Chapman and Hall/CRC.

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: Series b (Statistical Methodology)*, **71**(2), 319–392.

Schlather M, Malinowski A, Menck PJ, Oesting M, Strokorb K (2015). "Analysis, Simulation and Prediction of Multivariate Random Fields with Package **RandomFields**." *Journal of Statistical Software*, **63**(8), 1–25. URL http://www.jstatsoft.org/v63/i08/.

Schlather M, Malinowski A, Oesting M, Boecker D, Strokorb K, Engelke S, Martini J, Ballani F, Moreva O, Auel J, Menck PJ, Gross S, Ober U, Ribeiro P, Ripley BD, Singleton R, Pfaff B, R Core Team (2019). **RandomFields**: *Simulation and Analysis of Random Fields.* R package version 3.3.6, URL https://cran.r-project.org/package=RandomFields.

Simpson D, Lindgren F, Rue H (2012). "Think Continuous: Markovian Gaussian Models in Spatial Statistics." *Spatial Statistics*, **1**, 16–29.

Stein ML (2014). "Limitations on Low Rank Approximations for Covariance Matrices of Spatial Data." *Spatial Statistics*, **8**, 1–19.

Sun Y, Li B, Genton MG (2012). "Geostatistics for Large Datasets." In *Advances and Challenges in Space-Time Modelling of Natural Events*, pp. 55–77. Springer.

Varin C, Reid N, Firth D (2011). "An Overview of Composite Likelihood Methods." *Statistica Sinica*, **21**, 5–42.

Vecchia AV (1988). "Estimation and Model Identification for Continuous Spatial Processes." *Journal of the Royal Statistical Society: Series B (Methodological)*, **50**(2), 297–312.

Veness C (2010). "Calculate Distance, Bearing and More Between Latitude/Longitude Points." *Movable Type Scripts*, pp. 2002–2014.

Vetter JS (2013). *Contemporary High Performance Computing: From Petascale toward Exascale.* Chapman and Hall/CRC.

West M (1987). "On Scale Mixtures of Normal Distributions." *Biometrika*, **74**(3), 646–648.

Wickham H (2016). **ggplot2**: *Elegant Graphics for Data Analysis.* Springer-Verlag New York. ISBN 978-3-319-24277-4. URL http://ggplot2.org.

Xu G, Genton MG (2017). "Tukey g-and-h Random Fields." *Journal of the American Statistical Association*, **112**(519), 1236–1249.

Affiliation:

Sameh Abdulah
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: sameh.abdulah@kaust.edu.sa

Yuxiao Li
Statistics Program (STAT)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: yuxiao.li@kaust.edu.sa

Jian Cao
Statistics Program (STAT)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: jian.cao@kaust.edu.sa

Hatem Ltaief
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: hatem.ltaief@kaust.edu.sa

David E. Keyes
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: david.keyes@kaust.edu.sa

Marc G. Genton
Statistics Program (STAT)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: marc.genton@kaust.edu.sa

Ying Sun
Statistics Program (STAT)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia, 23955-6900
E-mail: ying.sun@kaust.edu.sa