blitzKrig: An R Package for Fast Geostatistics with Kronecker Covariances

by Dean Koch, Subhash Lele, and Robert Crabtree

Abstract We introduce in this paper a new R package, called **blitzKrig**, for modelling two-dimensional stationary Gaussian Processes. The package assumes gridded data and a special covariance structure wherein the joint covariance matrix V can factored using Kronecker products. This simplifies several important modelling equations, including the likelihood function, the generalized least squares equation, and the kriging predictor and variance equations. Our package offers computationally lean implementations of these equations (and more). We demonstrate its kriging functionality with predictions on the Meuse soils datasets, and also report on a benchmark experiment for computation time that shows improvements of several orders of magnitude compared to four major alternative R packages for spatial kriging.

Introduction

The second-order stationary Gaussian Process (SGP) plays a central role in modern analysis of spatial data. In geostatistics it underlies several important techniques of inference and prediction (Chiles and Delfiner 2012), including generalized least squares and kriging. We see it also in engineering and computer experiments with surrogate models (Gramacy 2016), in machine learning prediction algorithms (Rasmussen and Williams 2006), and in probabilistic models for ecological systems (Koch, Lewis, and Lele 2021), among many other applications.

In the SGP model, the joint probability distribution for n points z_k, with coordinates x_k, y_k, is multivariate Gaussian, and its variance-covariance matrix V is generated by a function C of the separation vector between points $V_{ij} = C(x_i - x_j, y_i - y_j)$ (Cressie 2015). One of the charms of SGP theory is the simplicity in analytic expressions for things like likelihood, conditional expectation, and least squares estimates, all of which involve V and its sub-matrices.

However, V is a major headache for computer programmers. The number of entries in this matrix is n^2, so as sample sizes grow large, seemingly routine matrix expressions become surprisingly difficult to evaluate. For large enough n, readers will find that either V is too large to fit in computer memory, or the order-n^3 complexity of its factorization is prohibitively slow, or both (Lindgren, Rue, and Lindström 2011).

These are major obstacles to software implementations of SGP, particularly when it comes to kriging. Kriging workflows with popular packages like **gstat**, **geoR**, and **fields** can become so slow as to be unworkable (or even fail entirely due to out-of-memory errors) when the number of point locations of interest are large in number, as we will see in the Computations section.

This practical upper limit on sample sizes with conventional models motivates us to introduce a new package, **blitzKrig**, whose implementation of the SGP is designed around Kronecker covariances on grid data, for improved computational efficiency. The computational ceiling is still there, but it is extended much higher. **blitzKrig** offers:

- Optimized functions for likelihood, GLS, kriging, and simulation
- Flexible parametric covariance structures supporting anisotropy
- Simple beginner-friendly workflows for down-scaling
- Compatibility with data objects from **raster**, **terra**, and **sf** packages

The likelihood function is particularly important here because it will enable modelers to adapt the methods in **blitzKrig** as an engine for handling autocorrelation in more complex nonlinear spatial models, such as in the simulation of integro-difference equations. In fact, this package is built from code that was originall written for such a model, in a study on mountain pine beetle outbreaks (Koch, Lele, and Lewis 2020).

We illustrate the speed-up offered by **blitzKrig** in the Computations section, with a comparison of computation times on a variety of ordinary kriging problems. This follows the Kriging Example section, where we present a detailed demonstration of a universal kriging workflow on the Meuse soils dataset. First we review some related tools and introducing the modelling approach that makes **blitzKrig** unique.

Table 1: A list of one-dimensional correlation functions available in blitzKrig. Kronecker covariances are constructed from the product of a pair of these functions, one receiving the x separation distance as its argument, and the other the y distance. Normalization constants are omitted for brevity, and K_p denotes the order-p Bessel function of the second kind (where p is a shape parameter).

code	name	alias	$c(\Delta)$
exp	Exponential	-	$\exp(-\Delta)$
gau	Gaussian	Squared-Exponential, or Stable Kernel	$\exp(-\Delta^2)$
gex	Gamma-Exponential	Power-Exponential	$\exp(-\Delta^p)$
mat	Matérn	Whittle-Matérn	$\Delta^p K_p(\Delta)$
sph	Spherical	-	$1-(3/2)\Delta+(1/2)\Delta^3$

Background

One strategy for computation with V on large-n problems is parallelization. The **bigGP** (Paciorek et al. 2015) and the **laGP** (Gramacy 2016) packages, for example, distribute the task of block factorization to multiple linked processes. This can speed likelihood calculations by several orders of magnitude in large-n examples.

However, these packages have steep learning curves, and are intended for high-memory, many-core computers. **blitzKrig** instead aims to be simple to learn and operate, with a memory footprint suitable for use on ordinary desktop computers.

Large-n solutions of this type usually make use of local approximations to the desired covariance function, with the goal of introducing sparsity in V or its inverse (the precision matrix). Examples include: covariance tapering and fixed rank kriging, like in **LatticeKrig** (Nychka et al. 2016) and **FRK** (Zammit-Mangion and Cressie 2021); Markov Random Field approximations (Lindgren, Rue, and Lindström 2011), and Bayesian approximations like R-INLA (Lindgren and Rue 2015), **laGP** (Gramacy 2016), and **spBayes** Finley, Banerjee, and E.Gelfand (2015).

The approach in **blitzKrig** is unusual in that it computes exact likelihood (and kriging predictions) but also supports long-tailed (non-compact) covariance functions, including the very common Gaussian covariance function. The dense covariance matrices V that result from this choice would normally be problematic, but our package does not rely on sparsity.

Instead it uses an algebraic shortcut that emerges for gridded layouts and certain covariance functions, called *Kronecker covariances* (Koch, Lele, and Lewis 2020; Drton, Kuriki, and Hoff 2021). This shortcut also pops up in auto-regression (Martin 1979) and separable space-time SGPs (Genton 2007), but despite its elegant computational properties (Van Loan 2000) we rarely see it used in purely spatial problems like ordinary kriging.

CRAN lists a number of alternatives for fitting exact SGP models (R. Bivand and Nowosad 2022), but three stand out for their scope, maturity, and quality of documentation: The **gstat** package for variogram-based geostatistical modeling (E. J. Pebesma 2004; R. S. Bivand, Pebesma, and Gómez-Rubio 2013); The **geoR** package, which supports variogram, likelihood, and Bayesian techniques (Ribeiro Jr and Diggle 1999; Diggle and Ribeiro 2007), and **fields** a feature-rich interpolation package by Douglas Nychka et al. (2021) for SGPs and spline models.

The RandomFields package by Schlather et al. (2015) also deserves mention, but is no longer in active development and not currently listed on CRAN. We included the most recently archived version of this package (2022-05-04) in the comparisons of the Computations section, along with the most recent major release of **gstat**, **geoR**, and **fields** (as of 2022-09-04).

Model

blitzKrig models a response data vector z of point values z_k (for $k = 1, \ldots n$) as an observation of the n-dimensional random vector $Z \sim \mathrm{N}(X\beta, V)$. This splits z into a deterministic trend component and a random spatial component.

The trend is represented by the expected value $X\beta$, where X is a known covariate data matrix and β an unknown vector of coefficients. The spatial component is represented by a two-dimensional (2d) covariance function C which generates the covariance matrix V.

In **blitzKrig**, the covariance function is constructed using the product of two one-dimensional components c_x and c_y. These are functions of the x and y component distances, Δ_x and Δ_y, and the covariance function has the form:

$$C(\Delta_x, \Delta_y) = \sigma^2 c_x(\Delta_x)\, c_y(\Delta_y) + \epsilon 1_{\{\Delta_x = \Delta_y = 0\}}. \tag{1}$$

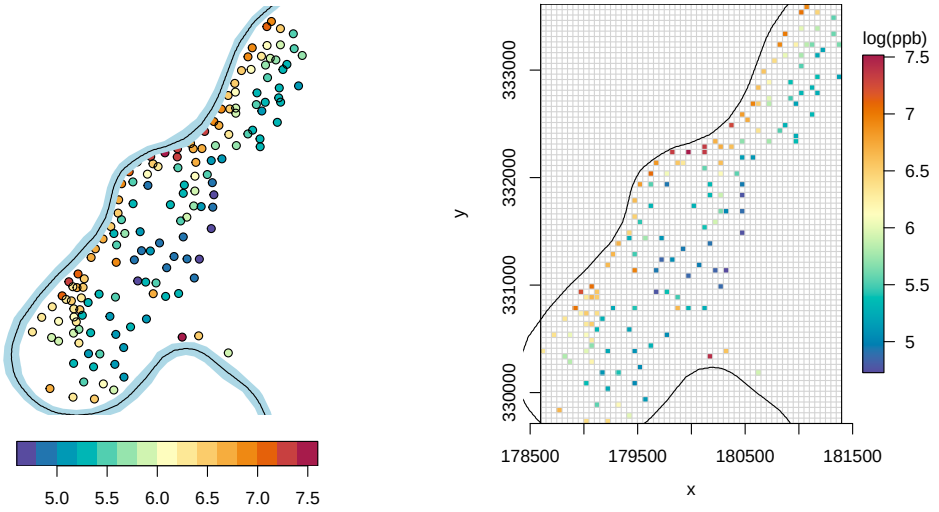

Figure 1: Zinc concentrations (in log parts per billion) from the Meuse dataset, a soil survey on the floodplain of the Meuse River (left). These data are snapped to a grid for use with blitzKrig (right)

where component distances are scaled by range parameters ρ_x and ρ_y,

$$\Delta_x = \frac{|x_i - x_j|}{\rho_x} \quad \text{and} \quad \Delta_y = \frac{|y_i - y_j|}{\rho_y}. \tag{2}$$

We call this a *Kronecker covariance* because when the observed data lie on a regular grid (and we assume they do), the matrix $V - \epsilon I$ takes the form of a Kronecker product. Any pair of correlation functions can be used, and their parameters may take on different values in the x and y directions. Table 1 lists all functional forms for c currently implemented in **blitzKrig**.

Parameters σ and ϵ are the partial sill, and nugget variance, respectively. The nugget variance can be understood here to represent aspatial measurement error, whereas the partial sill, and range parameters describe the variance of points and the spatial profile of correlations between them.

For a more complete description these parameters and their various interpretations, see Cressie (2015). In particular, it is helpful to understand their role in defining a graphical diagnostic known as the semi-variogram curve, some examples of which are found towards the end of the Kriging Example section.

Kriging Example

blitzKrig is primarily a kriging package, so we will demonstrate its core features in a point interpolation problem. Those familiar with the documentation for the **sp** and **gstat** packages will recognize the Meuse dataset, which appears in many R code examples and in the vignette by E. Pebesma (2022).

This is a survey of soil heavy metal concentrations the Meuse river floodplain in the Netherlands, introduced by Burrough, McDonnell, and Lloyd (2015) and later reproduced as part of the **sp** package by E. J. Pebesma and Bivand (2005). In this example we will interpolate zinc concentration while adjusting for a linear covariate, distance to river.

The SGP is a model for a multivariate normal random variable, so users should always begin by considering whether their data fits this assumption closely enough. In our case, log-transforming the concentration data produces a response variable that more closely resembles a sample from the expected distribution. These log-zinc values been loaded already in the **sf** points data-frame named `pts`. Figure 1 (left) shows how the 155 observations are positioned relative to the river.

```
# snap log zinc data to grid of specified resolution
g = bk_snap(pts, g=list(gres=c(y=50, x=50)))

#> maximum snapping distance: 33.2640947569598
```

blitzKrig only supports points lying on a regular grid. Irregularly spaced points like the Meuse data must be first snapped to such a grid. This is a matter of selecting a resolution sufficiently fine

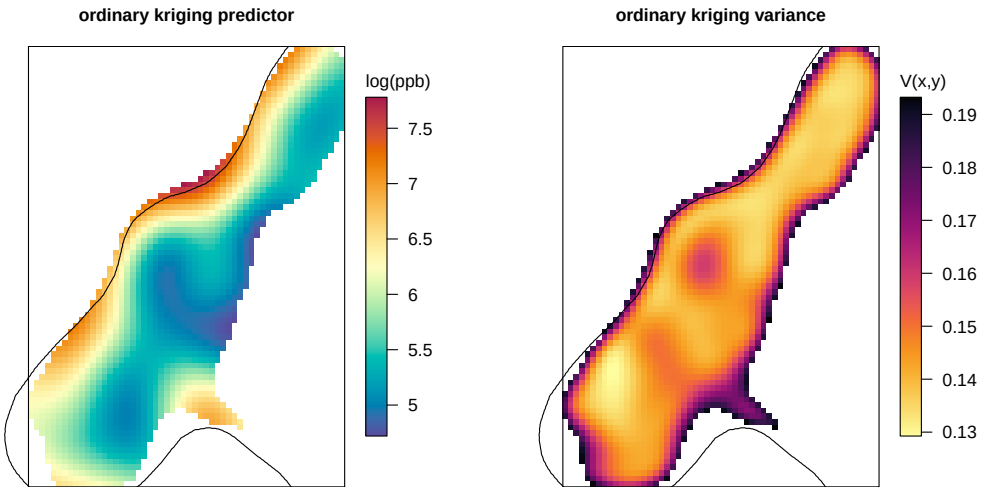

Figure 2: Ordinary kriging prediction and variance heatmaps generated by blitzKrig for the Meuse example. Predictions are generated for the entire grid, but are masked here to show detail in areas nearest the observed points.

as to produce an acceptably small positional error. In the code above, we used `bk_snap` to specify a resolution of 50 x 50 metres, creating a grid of dimensions 78 x 56 (rows x columns) with extent covering the Meuse point sample (Figure 1, right). At this resolution most positional errors are in the range of 10-25 metres distance, which is good enough for the demonstration here.

Ordinary Kriging

The new gridded version of the data is now in the format expected by the **blitzKrig**'s core modelling functions. A covariates data matrix `X` may optionally be supplied, as we do later, to account for linear trends. First, by way of comparison, we will ignore covariates and fit a spatially constant trend.

`bk_fit` attempts to automatically find maximum likelihood estimators (MLEs) for the mean and covariance parameters by numerically optimizing the full joint likelihood for all observed points in `g`, using R's `base::optim`.

```
# fit the covariance model and mean
fit_result_ok = bk_fit(g, quiet=TRUE)
```

The default covariance function implemented in `bk_fit` is the isotropic Gaussian, but a variety of alternatives are available (see Table 1 and the Model section). Sensible defaults for initial values and bounds are set automatically based on the sample variance and grid dimensions.

To interpolate observed data, pass the covariance parameters (argument `pars`) and a grid containing the observed data (argument `g_obs`) to `bk_cmean`. This populates all grid points in `g_obs` with values from the kriging prediction equation (including at observed points). Variance is computed separately, but the calling syntax is the same apart from argument `out='v'`.

```
# compute conditional mean and variance
z_ok = bk_cmean(g_obs=g, pars=fit_result_ok[['pars']])
z_ok_var = bk_cmean(g_obs=g, pars=fit_result_ok[['pars']], out='v', quiet=TRUE)
```

Figure 2 displays the output, masked to a neighbourhood of the observed data. Here we have requested predictions over the same grid that was used for fitting. Users can request any output grid they like, by snapping the observed data to it and passing it to `bk_cmean` in argument `g_obs`. At the end of this section we will predict over a grid at much finer resolution.

The workflow up to this point has been ordinary kriging (OK), as we did not use any covariates to adjust for linear trends. We will do that now, by including distance to river (along with its square root) as example predictors, and re-fitting the model to demonstrate universal kriging (UK).

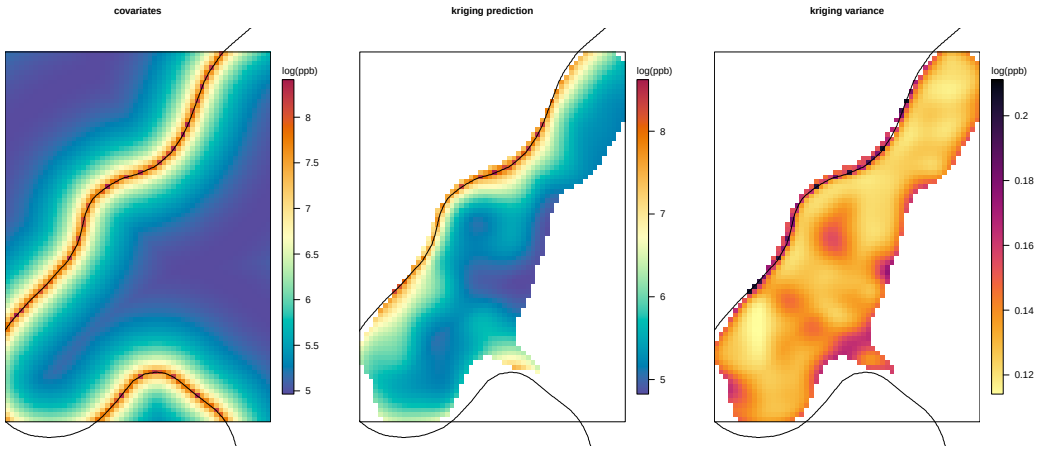

Figure 3: Universal kriging predictions and variance generated by blitzKrig for the Meuse example. The response variable (log zinc concentration) is de-trended using a linear predictor (left) based on distance to river and its square root during model fitting, resulting in more detail in kriging predictions (middle), and a decrease in kriging variance (right)

Universal Kriging

To include covariates in a model fit, simply pass a matrix `X` of covariate values at the observed point locations in your call to `bk_fit`. For prediction, the `X` argument in `bk_cmean` should include all of the output grid point locations.

Covariate data (rows) must be supplied in the same order as the response data in `g`. In the code below we construct the full matrix `X` by using `bk_coords` to export the grid point locations to an `sf` points data-frame in the correct order, then use `sf::st_distance` to compute distances to a line geometry representing the middle of the river.

```
# measure distances for every point in the grid
river_dist = sf::st_distance(bk_coords(g, out='sf'), meuse_list[['river_line']])

#> processing 4368 grid points...

# include both distance and its square root
river_dist = units::drop_units(river_dist)
X = scale(cbind(river_dist, sqrt(river_dist)))

# find the subset of predictors at observed response locations
is_obs = !is.na(g[['gval']])
```

The UK workflow can now proceed like OK, except with `X` passed to `bk_fit` and `bk_cmean`.

```
# fit the covariance model again with X
fit_result_uk = bk_fit(g_obs=g, X=X[is_obs,], quiet=TRUE)

# compute conditional mean and variance (supply all distances in X this time)
z_uk = bk_cmean(g, fit_result_uk[['pars']], X=X)
z_uk_var = bk_cmean(g, fit_result_uk[['pars']], X=X, out='v', quiet=TRUE)
```

These functions account for `X` by calling `bk_GLS` automatically in the course of calculations to de-trend the response, z. Given a set of MLE covariance parameters, `bk_GLS` estimates the trend $X\beta$ using the generalized least squares (GLS) expression for the effects vector $\hat{\beta}$.

```
# use GLS to estimate the spatially varying trend
z_lm = bk_GLS(g, fit_result_uk[['pars']], X=X, out='z')
```

We called `bk_GLS` by hand in the code above to produce the image in the left pane of Figure 3, showing the values $X\hat{\beta}$ over the entire output grid. The middle and right panes show the resulting UK predictions and variance.

Diagnostics

Assuming the covariance model is a good fit to the data, kriging theory guarantees that `bk_cmean` will return the best linear unbiased predictor for grid `g_obs`, in the sense of minimizing kriging variance.

To find the best fitting model for their data, users are encouraged to seek out informative covariates to include in `X`, and to check model fitting diagnostics for problems that would impact predictions. **blitzKrig** provides two visual diagnostics in the functions `bk_plot_pars` and `bk_plot_semi`.

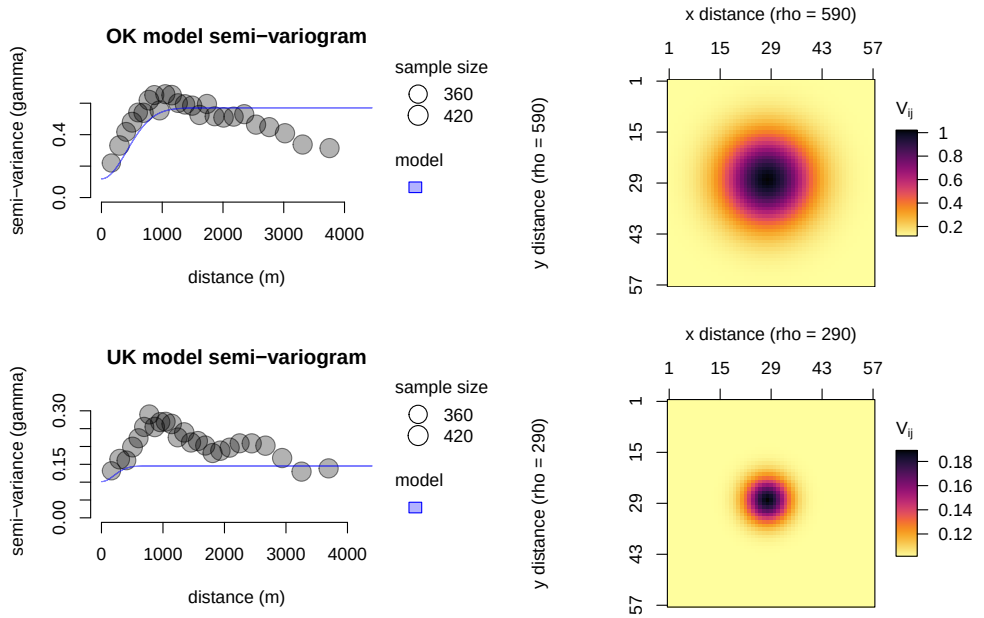

Figure 4: Fitted covariance models from kriging on the Meuse dataset, visualized in two ways: On the left, estimated semi-variogram values (circles) are plotted next to model predictions (blue curves). On the right, a heatmap displays covariances with respect to the central point. The top two plots show the fitted OK model. The bottom two plots show the UK model, where removing a linear trend from the response data has resulted in lower variance and a smaller range.

The output of `bk_fit` includes a list of covariance parameters. Any such parameter list can be visualized using `bk_plot_pars`. This shows the footprint of covariances between any given point and its surrounding points and indicates the level of smoothing to be expected in predictions. Another type of visualization, the semi-variogram, can be plotted with `bk_plot_semi`. Various estimators for the semi-variogram are implemented in `bk_sample_vg`. These generate a point cloud of sample semi-variances, estimated directly from the data, that can be compared visually with the theoretical curve obtained via MLE.

Diagnostic plots from the OK and UK workflows are shown together for comparison in Figure 4. Accounting for distance to the river has a dramatic effect on the fitted covariance model. The UK model estimates a much lower variance than OK, and favors a much smaller effective range (or spatial scale of correlations). This leads to more spatial detail in the kriging predictions from UK.

The initial values and bounds used in fitting covariance parameters can be found in the output of `bk_fit` (data-frame `bds`). We encourage users to understand these settings, and to be on the lookout for common problems with fitting, such as parameters converging to a bound, or not moving from their initial values. Both can indicate problems of model-misspecification, or simply a poor choice of initial values.

Anistropic Models

Another way of improving model fit is to explore covariance functions with more flexibility than the default isotropic Gaussian. The term *isotropic* refers to radial symmetry in C (equal distance implying equal covariance), and *anisotropic* refers to its absence. **blitzKrig** supports models of both kinds.

The following code fits two examples to the Meuse data; The first is a Gaussian covariance with anisotropy, and the second a Matérn product covariance (Koch, Lele, and Lewis 2020).

```
# fit a 2 + 2 parameter Gaussian covariance with anisotropy
```

```
fit_result_uk_gau = bk_fit(g_obs=g, X=X[is_obs,], iso=FALSE, quiet=TRUE)

# fit a 2 + 4 parameter product Matern
fit_result_uk_mat = bk_fit(g_obs=g, X=X[is_obs,], pars='mat', iso=FALSE, quiet=TRUE)
```

The `iso=FALSE` argument allows c_x and c_y to take on different parameters, whereas the default `iso=TRUE` forces $c_x = c_y$. The `pars='mat'` argument is shorthand for `pars=c(y='mat', x='mat')`. It specifies that c_x and c_y should both be Matérn functions.

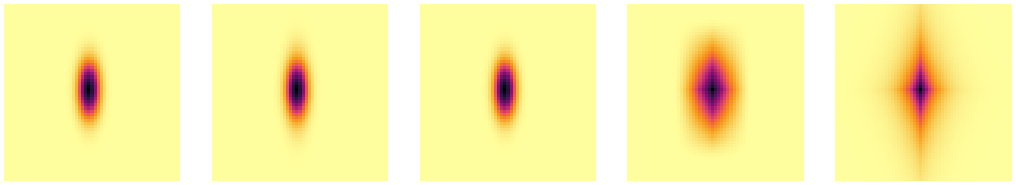

Figure 5: Five examples of anisotropic covariance structures fitted to the Meuse data. As in the previous figure, darker pixels indicate stronger correlation with the central point. From left to right, these are the Kronecker covariances formed by setting both c_x and c_y equal to "gau", "mat", "gxp", "sph", and "exp" models, respectively.

The anisotropic Gaussian function has one more parameter than its isotropic counterpart, since it allows the two range parameters ρ_x and ρ_y to vary independently. The Matérn product function adds two more shape parameters, controlling the degree of kurtosis in two directions. The fitted parameters from these two models are visualized as the first two heatmaps on the left of Figure 5, together with models for three other choices of `pars`.

The Gaussian is the only isotropic model in **blitzKrig**. The product Matérn function (`fit_result_uk_mat` above), for example, does *not* generalize the more common 2d (isotropic) Matérn function. It does however approach the Gaussian as $p \to \text{inf}$ (Koch, Lewis, and Lele 2020).

Other choices of c_x, c_x lead to a variety of other types of anisotropy, but always with a directionality oriented along one or both of the coordinate axes. Differently oriented data could be modeled by estimating the direction of anisotropy in a preliminary analysis (*eg.* by the method of Koch, Lele, and Lewis 2020) then rotating the observed point locations by this angle prior to snapping. For simplicity, we skip this step here.

Anisotropic covariance functions impart complexity, in the form of new parameters, but also flexibility. Flexibility is a double-edged sword. It can improve a model by more closely aligning it with reality, or it can worsen it by enabling over-fitting, which leads to underestimates of uncertainty and poor predictions.

Cross-validation

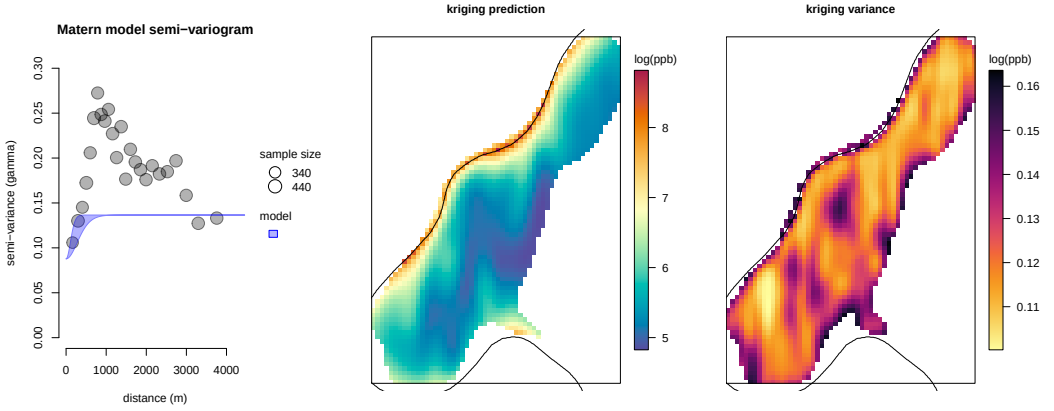

Figure 6: Universal kriging results for the Meuse example using a product Matérn covariance function. The semi-variogram (left) is now a ribbon plot, showing a range values at a given distance due to anisotropy. The middle and right panes show the model predictions and their variance.

Table 2: Estimates of the root mean squared prediction error on the Meuse dataset for log zinc (rMSPE) and its back-transformed values (rMSPEb) in a 25 X 5-fold cross-validation (CV) experiment. Results are reported for the four kriging models presented earlier, illustrating a progression of improvement as we add covariates and refine the covariance model.

name	covariance	isotropic	covariates	parameters	rMSPE	rMSPEb
fit_result_ok	gau x gau	TRUE	0	5	0.3147218	312.3033
fit_result_uk	gau x gau	TRUE	2	7	0.2872748	160.6523
fit_result_uk_gau	gau x gau	FALSE	2	8	0.2767657	156.5372
fit_result_uk_mat	mat x mat	FALSE	2	10	0.2641235	151.3025

Figure 6 shows the semi-variogram, predictions, and variance from the Matérn product model fitted in the previous section. The predictions have more spatial complexity compared to the isotropic UK model fitted earlier on, and variance has decreased slightly. Are we over-fitting? One way to check is through cross-validation (CV).

In CV, we withhold a subset of the data (a hold-out set, or fold) during model fitting, and then come back to it later as a reference to compare predictions. If we have an over-fitting problem, the model is likely to predict well on the training locations, but poorly on the hold-out locations. The hold-out error gives an indication of what to expect when predicting at an unseen locations.

We performed a 5-fold cross-validation experiment, by dividing the 155 observed Meuse points (randomly) into five folds, each of size n=31. For each fold, we fitted all four models to the complementary subset of the data (size n=124), then computed the mean squared prediction error on the hold-out set for log-zinc (MSPE) and its back-transformed version, MSPEb (see the Back-transforming section below).

As these statistics were sensitive to the choice of partition, we repeated the process 25 times and averaged the results. They are reported in Table 2. Lower scores indicate better predictive performance and a better fitting model. We found a progression of improvements as we added complexity: starting with our baseline model (`fit_result_ok`), then adding covariates (`fit_result_uk`), introducing anisotropy (`fit_result_uk_gau`) and, finally, introducing additional flexibility in covariance shape (`fit_result_uk_mat`).

Back-transforming

If predictions are required on the original scale (ppb, instead of its logarithm), users may think to simply take the exponential of the kriging predictions vector `z_uk`. However this leads to underestimates of the (random) variable $\exp(z)$ (by Jensen's inequality), so we recommend adding one half the kriging variance to `z_uk` before exponentiating (Cressie 2015). Note that this particular bias-adjustment is specific to the log-transformation.

Computations

This final section explores the computational bottlenecks that can make kriging slow, and that techniques that **blitzKrig** uses to circumvent them. The core idea is to rewrite the matrix $V - \epsilon I$ as a Kronecker product (see the Model section), so that operations involving sub-matrices of V can be simplified.

Two sub-matrices are particularly important in this context. The first is the $n_o \times n_o$ covariance V_o, for the n_o observed points. V_o has to be generated (its entries calculated), then factorized, so that products of the form $V_o^{-1}z$ can be calculated. This happens in many routine spatial statistical operations - including likelihood, GLS, kriging equations, and conditional simulation - and it the reason they become computationally intensive for large n_o.

The second important sub-matrix is the $n_p \times n_o$ cross-covariance V_p, between the observed locations and the n_p unseen prediction locations. n_p can be very large, particularly when a gridded output is needed, making products with V_p expensive both in terms of computation time and memory use. Such products appear in both the kriging prediction and variance equations.

blitzKrig generates the entries of V_o and V_p directly from the Kronecker product factorization of $V - \epsilon I$. This speeds construction time for V_o and V_p, and, more importantly, it enables the package to multiply a vector by V_p without having to explicitly build V_p in computer memory.

Benchmark Comparisons

To illustrate the speed of **blitzKrig** we timed computations for the Meuse OK exercise presented in the Kriging Example section, repeating it several times over a range of output resolutions (varying n_p) and recording median evaluation times in five repetitions using **microbenchmark**.

For comparison we also tested four popular R packages mentioned in the introduction, **gstat**, **geoR**, **fields**, and RandomFields, all using the same isotropic Gaussian model fitted with default settings. To get a range of observed sample sizes (n_o) values we repeated this process with five other point datasets, one on ozone concentration, and four on forest density.

The ozone data are included as the object `Chicago03` in the **fields** package (Douglas Nychka et al. 2021) and appear frequently in its example code. They are Environmental Protection Agency recordings of average ozone concentration in the air at $n_o = 20$ Chicago-area stations in 1987.

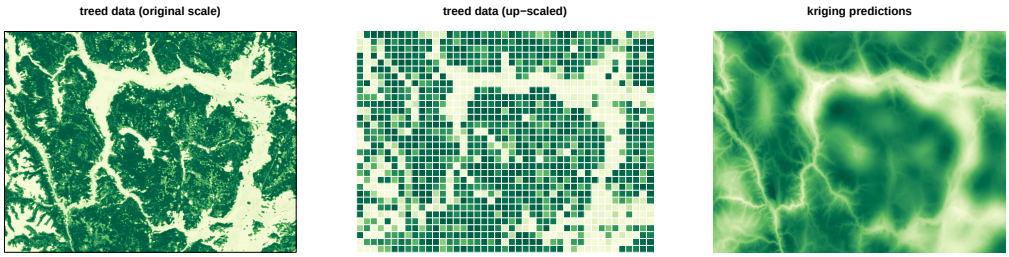

Figure 7: A 1021 x 1349 raster on forest density in central BC, Canada (left) is up-scaled to produce a much coarser resolution version with dimensions 32 x 43 (middle). blitzKrig can rapidly downscale rasters like these. In a UK model adjusting for elevation, the predictions at right (at original resolution) were generated in less than a second.

The forest data come from Canada-wide estimates of forest characteristics by Beaudoin et al. (2018). Estimates of areal tree cover (%) for the province of British Columbia (BC) are re-published in **rasterbc** as (raster) tiles, each containing around one million data points. We created 4 much smaller example datasets by taking sub-samples of points from a tile in Central BC (Figure 7, left).

For one of the forest sub-samples (which we call treed) we picked one thousand points at random. For the others we sampled points at regular intervals, so that the sub-samples themselves formed rasters. These up-scaled datasets are named treed_88, treed_352, treed_1376, with the suffix indicating n_o. Figure 7 (middle) shows the largest.

The three raster examples are special in that the observed point sets form complete grids. They contain no `NA`s. The three irregularly sampled point sets (ozone, Meuse, and treed) are examples of the incomplete data case. All but around 0.1% of the points in the treed grid are `NA`, and in the (snapped) ozone and Meuse grids, empty, un-sampled spaces are filled with `NA`s (see Figure 1, right).

Complete Data

This distinction between complete and incomplete is important because both $V - \epsilon I$ and $V_o - \epsilon I$ have Kronecker product factorizations in the complete case, and this makes V_o much easier to deal with. The result is big performance improvements in **blitzKrig** on large n_o problems, particularly in evaluations of the likelihood.

The improvement is illustrated in Figure 8, which plots evaluation time against n_o for likelihood-based model fitting. We excluded **gstat** from this comparison because it uses variograms (instead of MLE) to parametrize covariance. The other packages all follow a similar trajectory of rising computation times, with the exception of **blitzKrig** in the complete data case (dotted lines). Speedy likelihood function evaluations by **blitzKrig** led to faster fitting times for large n_o.

In a log-log plot like Figure 8, a linear trend with slope p reflects a complexity power law with exponent p. What is noteworthy here is in the large-n_o behavior, which shows a similar p for all packages except **blitzKrig** on complete examples, where it is faster than the rest by a factor that grows *exponentially* with n_o. These differences reflects the rate-limiting operation of factoring V_o. In **blitzKrig** its algorithmic complexity is $O(n_o^3)$ for incomplete data, and closer to $O(n_o^{3/2})$ for complete data.

The performance advantage of **blitzKrig** on complete grids is not limited to kriging-related problems. We can expect similar improvements in computation time for any application of likelihood, GLS, or simulations from SGPs. Note that the completeness requirement is strict - any number of `NA`s in the data grid means the grid is incomplete

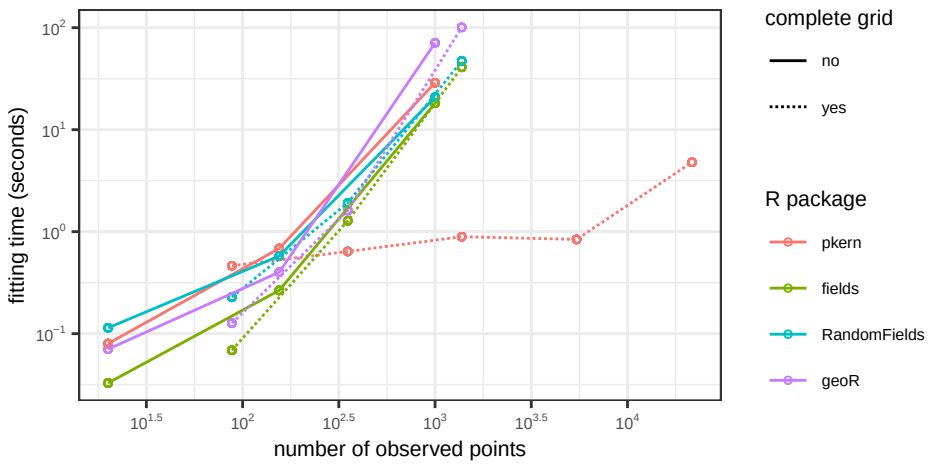

Figure 8: Evaluation times for likelihood-based OK model fitting to example data with a range of sample sizes (n_o). Cases where the observed data form a complete regular grid are indicated by dashed lines, whereas incomplete cases are indicated with solid lines. The likelihood function in blitzKrig is optimized for the complete case, leading to faster performance. Two additional up-scaled treed datasets (with n_o 5440 and 21632) were tested to show large-n_o behaviour in blitzKrig.

Prediction and Variance

After model fitting we timed evaluations of the kriging prediction and variance equations for ten prediction grids, ranging in size from 8×11 ($n_p = 88$) to 4084×5396 ($n_p > 22$ million). To limit the total length of the experiment, the function calls for each package were halted (timed-out) at around two minutes.

For **blitzKrig** and **fields**, we timed kriging prediction and variance separately. In **gstat** and **geoR**, both are returned from a single function call, so we only recorded the combined time. In RandomFields, variance was unavailable at the time of writing, so we recorded only prediction time.

The times are plotted in Figure 9 as a function of prediction grid size, n_p. Results from each data example are shown separately in two rows of three panels - the top three show results from the incomplete cases, and the bottom three from the complete cases. One thing that is obvious right away is the relative speed of prediction (dashed lines) compared to variance and prediction (solid lines). This is because in the variance equation we must multiply V_p by a matrix, whereas in prediction we multiply it by a vector (increasing complexity by a factor of n_o). This makes RandomFields, **fields**, and **blitzKrig** stand in out the large n_p results for having fast prediction-only methods.

Even with incomplete data, the prediction method of **blitzKrig** was the fastest by far on all but the smallest n_p and n_o examples. As we saw with likelihood, the relative speed-up increases with n_o, and by $n_o = 1000$ it is nearly two orders of magnitude faster than the next fastest method for large n_p problems. Variance was also faster with large n_o on the incomplete examples, but to a lesser extent.

With complete data, prediction times were even faster. For example, kriging from the treed_1376 points onto a 1021×1349 grid took less than half a second in **blitzKrig**, whereas the next fastest package (**fields**) required 90 seconds.

blitzKrig is naturally suited down-scaling with complete data, and the addition of covariates and anisotropy introduces little additional computational complexity in prediction. For example, it took less than three seconds to create the 1021×1349 UK output for the treed_1376 example in Figure 7 (right) using the anisotropic Gaussian covariance, and elevation, along with its square root, as covariates. Half of this time was used in fitting and half in prediction.

Variance computations were also much improved in the complete case, where the combined variance and prediction times for **blitzKrig** were faster than any other package by 1-2 orders of magnitude in all but the smallest n_p examples.

blitzKrig computes the variance in an unusual way (for both complete and incomplete problems). Instead of multiplying V_p by the $n_o \times n_o$ matrix V_o^{-1}, we multiply it be the eigen-vectors of V_o in a (length-n_o) loop, combining results as it goes. This is slower than matrix multiplication but it greatly reduces computer memory demands on problems with n_o and n_p both large, as we avoid ever having to write V_p or the product $V_p V_o^{-1}$ entirely in memory during kriging.

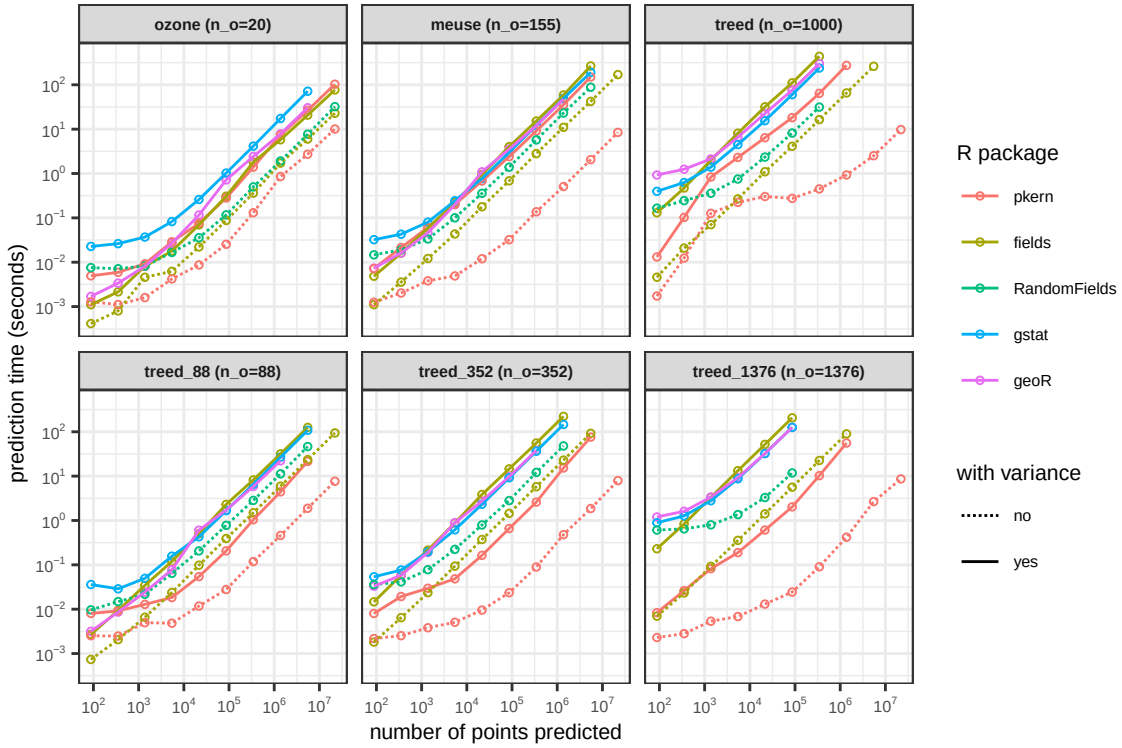

Figure 9: A comparison of computational performance in kriging as a function of the number of point prediction, n_p. Dashed lines indicate prediction time on its own, and solid lines indicate time to compute both predictions and variance. Panels separate results from six example, where top row examples are incomplete (grids containing NAs), and bottom row examples are complete. As n_o grows large, blitzKrig shows relatively fast performance, particularly on complete datasets.

Summary

Kronecker covariances are common enough in geostatistics. We see them everywhere in the form of Gaussian covariance functions, and separable space-time models. Nevertheless, we are unaware of any other CRAN R package for SGP models that factors the *spatial* covariance V into a Kronecker product, as we do in **blitzKrig**.

This factorization leads to huge computational advantages in evaluating likelihood and kriging equations (among other things), as we saw in Figures 8 and 9), with speed-ups of 1-3 orders of magnitude on problems with hundreds to thousands of observed points. This, for example, makes it feasible to downscale a raster onto a grid of several million points in seconds, using a statistically principled, MLE-based OK or UK model.

A fast interpolation tool like this could supplant methods like inverse distance weighting and Voronoi tiling, which, despite their bias issues, are often considered as reasonable fill-ins for when kriging is computationally impractical.

blitzKrig gets its performance edge through restrictions on covariance and point layout, but we nevertheless think it has wide applicability. Users can simply snap irregular points to a reasonably large grid, as we did in the Meuse example. And while some important covariance functions like the 2d Matérn are excluded, the alternatives provided will be flexible enough approximations in many cases (Koch, Lele, and Lewis 2020).

In specializing for Kronecker covariance, **blitzKrig** raises the upper limits for practical sample sizes and resolutions in SGP-based analysis, providing a more interactive and less frustrating user experience for analysts.

References

Beaudoin, A., P. Y. Bernier, P. Villemaire, L. Guindon, and X. Jing Guo. 2018. "Tracking Forest Attributes Across Canada Between 2001 and 2011 Using a k Nearest Neighbors Mapping Approach Applied to MODIS Imagery." *Canadian Journal of Forest Research* 48 (1): 85–93. https://doi.org/10.1139/

cjfr-2017-0184.

Bivand, R. S., E. Pebesma, and V. Gómez-Rubio. 2013. *Applied Spatial Data Analysis with r*. Use r! Springer New York.

Bivand, Roger, and Jakub Nowosad. 2022. "CRAN Task View: Analysis of Spatial Data," August. https://CRAN.R-project.org/view=Spatial.

Burrough, Peter A, Rachael A McDonnell, and Christopher D Lloyd. 2015. *Principles of Geographical Information Systems*. Oxford university press.

Chiles, JP, and P Delfiner. 2012. *Geostatistics: Modeling Spatial Uncertainty*. 2nd ed. Hoboken, NJ: John Wiley & Sons.

Cressie, Noel. 2015. *Statistics for Spatial Data*. John Wiley & Sons.

Diggle, P., and P. J. Ribeiro. 2007. *Model-Based Geostatistics*. Springer Series in Statistics. Springer New York. https://books.google.ca/books?id=qCqOm39OuFUC.

Douglas Nychka, Reinhard Furrer, John Paige, and Stephan Sain. 2021. "Fields: Tools for Spatial Data." Boulder, CO, USA: University Corporation for Atmospheric Research. https://github.com/dnychka/fieldsRPackage.

Drton, Mathias, Satoshi Kuriki, and Peter Hoff. 2021. "Existence and Uniqueness of the Kronecker Covariance MLE." *The Annals of Statistics* 49 (5): 2721–54.

Finley, Andrew O., Sudipto Banerjee, and Bradley P. Carlin. 2007. "spBayes: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models." *Journal of Statistical Software* 19 (4): 1–24. https://www.jstatsoft.org/article/view/v019i04.

Finley, Andrew O., Sudipto Banerjee, and Alan E.Gelfand. 2015. "spBayes for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models." *Journal of Statistical Software* 63 (13): 1–28. https://www.jstatsoft.org/article/view/v063i13.

Genton, Marc G. 2007. "Separable Approximations of Space-Time Covariance Matrices." *Environmetrics* 18 (7): 681–95.

Gramacy, Robert B. 2016. "laGP: Large-Scale Spatial Modeling via Local Approximate Gaussian Processes in r." *Journal of Statistical Software* 72 (1): 1–46. https://doi.org/10.18637/jss.v072.i01.

Koch, Dean, Subhash Lele, and Mark A Lewis. 2020. "Computationally Simple Anisotropic Lattice Covariograms." *Environmental and Ecological Statistics* 27 (4): 665–88.

Koch, Dean, Mark A Lewis, and Subhash Lele. 2021. "The Signature of Endemic Populations in the Spread of Mountain Pine Beetle Outbreaks." *Bulletin of Mathematical Biology* 83 (6): 1–24.

Koch, Dean, Mark Lewis, and Subhash Lele. 2020. "A Unifying Theory for Two-Dimensional Spatial Redistribution Kernels with Applications in Population Spread Modelling." *Journal of the Royal Society Interface* 17 (170): 20200434.

Lindgren, Finn, and Håvard Rue. 2015. "Bayesian Spatial Modelling with r-INLA." *Journal of Statistical Software* 63: 1–25.

Lindgren, Finn, Håvard Rue, and Johan Lindström. 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): 423–98.

Martin, RJ. 1979. "A Subclass of Lattice Processes Applied to a Problem in Planar Sampling." *Biometrika* 66 (2): 209–17.

Nychka, Douglas, Dorit Hammerling, Stephan Sain, and Nathan Lenssen. 2016. "LatticeKrig: Multiresolution Kriging Based on Markov Random Fields." Boulder, CO, USA: University Corporation for Atmospheric Research. https://doi.org/10.5065/D6HD7T1R.

Paciorek, Christopher J., Benjamin Lipshitz, Wei Zhuo, Prabhat, Cari G. Kaufman, and Rollin C. Thomas. 2015. "Parallelizing Gaussian Process Calculations in R." *Journal of Statistical Software* 63 (10): 1–23. https://doi.org/10.18637/jss.v063.i10.

Pebesma, Edzer. 2022. "The Meuse Data Set: A Brief Tutorial for the Gstat r Package." ViennaR.

Pebesma, Edzer J. 2004. "Multivariable Geostatistics in S: The Gstat Package." *Computers & Geosciences* 30: 683–91.

Pebesma, Edzer J., and Roger S. Bivand. 2005. "Classes and Methods for Spatial Data in R." *R News* 5 (2): 9–13. https://CRAN.R-project.org/doc/Rnews/.

Rasmussen, Carl Edward, and Christopher KI Williams. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT press.

Ribeiro Jr, Paulo J, and Peter J Diggle. 1999. "geoS: A Geostatistical Library for s-PLUS." *Technical Report*, 1–6.

Schlather, Martin, Alexander Malinowski, Peter J. Menck, Marco Oesting, and Kirstin Strokorb. 2015. "Analysis, Simulation and Prediction of Multivariate Random Fields with Package RandomFields." *Journal of Statistical Software* 63 (8): 1–25. http://www.jstatsoft.org/v63/i08/.

Van Loan, Charles F. 2000. "The Ubiquitous Kronecker Product." *J Comput Appl Math* 123 (1): 85–100.

Zammit-Mangion, Andrew, and Noel Cressie. 2021. "FRK: An r Package for Spatial and Spatio-Temporal Prediction with Large Datasets." *Journal of Statistical Software* 98 (4): 1–48.

Dean Koch
University of Alberta
Department of Mathematical and Statistical Sciences
11324 89 Ave NW, Edmonton, AB, Canada, T6G 2J5.
https://github.com/deankoch/blitzKrig
ORCiD: 0000-0002-8849-859X
dkoch@ualberta.ca

Subhash Lele
University of Alberta
Department of Mathematical and Statistical Sciences
11324 89 Ave NW, Edmonton, AB, Canada, T6G 2J5.
slele@ualberta.ca

Robert Crabtree
Yellowstone Ecological Research Center
Bozeman, MT
crabtree@yellowstoneresearch.org