Generalized Wendland Function

Thomas Caspar Fischer

May 13, 2022

1 Introduction

Kriging models are an important tool for geostatistical modeling. However, technological advances have greatly increased data volumes, to the point where the model lacks the scalability to work with these large data sets. This is in part because the Matérn covariance, which is commonly used in geostatistics because of its continuous parameterization of the smoothness, does not have compact support and will thus generally produce dense covariance matrices. For large datasets, computing the cholesky decomposition of the resulting covariance matrix becomes an intractable task, as the computational complexity of the operation is of order $\mathsf{O}(n^3)$. Attempts to address this issue have focused on sparse methods, proposing different ways to compose covariance functions with compact support and continuous smoothness parameters, for example by applying a taper to the Matérn covariance function, because there exist few covariance functions with both compact support and continuously parameterized smoothness of the underlying Gaussian Random Field.

One exception is the generalized Wendland correlation function, which is a function of distance with three parameters: a range (β) and two shape parameters (κ and μ). The function is positive definite if $\beta > 0,\ \kappa > 0,\ \mu \geq 1 + d/2 + \kappa$ (Bevilacqua et al., 2019). The shape parameter of the Matérn covariance and parameter κ of the generalized Wendland function follow the relationship $\nu = \kappa + 0.5$ (Bevilacqua et al., 2019). Yet unlike the Matérn covariance, the generalized Wendland function has compact support and has an additional parameter for smoothness. Note that the generalized Wendland covariance function approximates the Matérn covariance in the limit Bevilacqua et al. (2022).

The primary disadvantage that has so far prevented a wider adoption of the generalized Wendland covariance in geostatistics is its computational intensity, as it generally requires the use of numerical integration techniques to evaluate because the function does not generally have a closed form representation. We consider the following correlation with range one:

$$\rho_{1,\kappa,\mu}(h) = \begin{cases} \frac{1}{\mathsf{B}(1+2\kappa,\mu)} \int_h^1 (u^2-h^2)^\kappa\ (1-u)^{\mu-1} du & 0 < h < 1 \\ 0 & h \geq 1 \end{cases}$$

$$\rho_{1,\kappa,\mu}(h) = \begin{cases} \frac{1}{\mathsf{B}(2\kappa,\mu+1)} \int_h^1 u(u^2-h^2)^{\kappa-1}\ (1-u)^{\mu} du & 0 < h < 1 \\ 0 & h \geq 1 \end{cases}$$

There exist some special cases such as the Askey function, which arises for $\kappa = 0$, and others at $\kappa = 1, 2, \ldots$.

κ	$\rho_{1,\kappa,\mu}(h)$
0	$(1-h)_+^{\mu}$
1	$(1-h)_+^{\mu+1}(1+h(\mu+1))$
2	$(1-h)_+^{\mu+2}(1+\frac{h}{3}(\mu+2)+h^2(\mu^2+4\mu+3))$

with

$$\rho_{1,\kappa,\mu}(h) = (1-h)_+^{\mu} = \begin{cases} (1-h)^{\mu} & 0 < h < 1 \\ 0 & h \geq 1 \end{cases}$$

The covariance function corresponding to these correlation functions is then given by the expression

$$\phi_{\beta,\sigma,\kappa,\mu,\theta}(h) = \begin{cases} \sigma + \theta & h/\beta < \epsilon \\ \sigma\rho_{\beta,\kappa,\mu}(h/\beta) & 0 < h/\beta < 1 \\ 0 & h/\beta \geq 1 \end{cases}$$

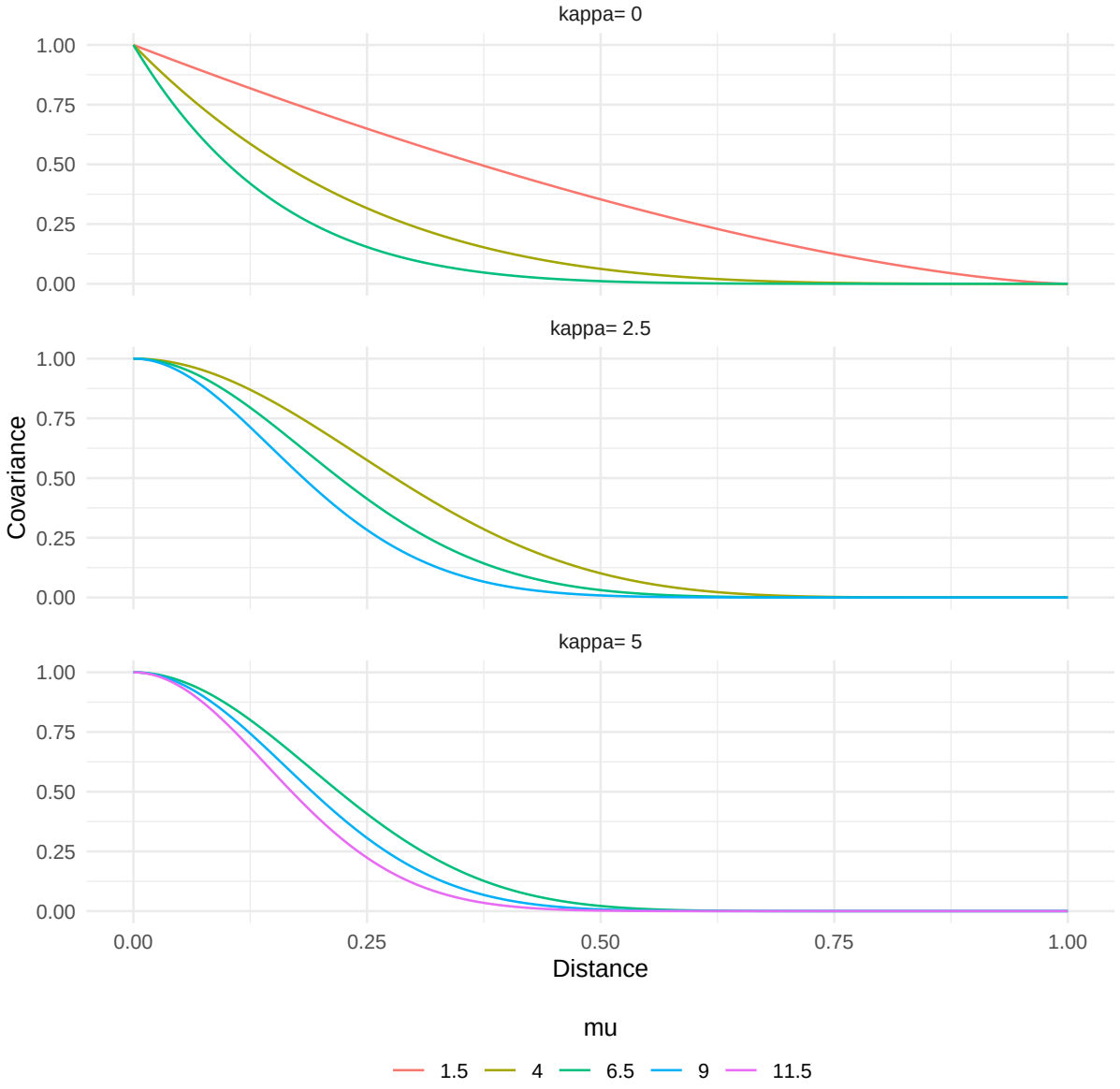


Figure 1: Wendland correlation function for different values of κ and μ.

This package intended to solve both of these issues. The covariance function implemented in this package reduces computation time by using a C++ backend, and further offers powerful approximation methods.

2 Installation

Users must install the GSL C library (Galassi et al., 2021) manually on their system. On unix-based systems the library can be directly installed from a source package or via `sudo apt install gsl-bin`. Windows users, on the other hand, will first need to install the Rtools toolchain (CRAN, 2022), locate and launch the `msys2` executable, run `pacman -Syu` to update the toolchain, and finally run `pacman -S mingw-w64-gsl` to install GSL.

Note that the GSL library is also a dependency of the **RcppGSL** package (Eddelbuettel and Francois, 2022), which will tell the user during installation whether the library has been installed correctly.

3 Features and how to use them

Albeit its properties make it useful for geostatistical modeling, the lack of a closed form of the generalized Wendland function necessitates the use of computationally expensive numerical integration. The impact is even substantial enough as to overpower the expected benefits of sparsity. The implementation in this package therefore provides users with multiple methods for approximating the actual function with less computationally expensive numerical interpolation. The most important features of the package can be summarized as follows:

1. Fully parameterized implementation of the generalized Wendland covariance function.
2. Option to estimate models with fixed range and/or nugget parameters.
3. Approximation methods based on numerical interpolation.
4. Full compatibility with **spam** (Furrer et al., 2022).
5. Easy interface to **optimParallel** (Gerber and Furrer, 2019).

3.1 Fully parameterized implementation

As was discussed in the previous section, the implementation offered here allows users to freely specify parameters β, κ, μ. Recalling that the covariance is only positive definite for a subset of the parameter space, users can further configure whether to make use of the parameterization $\mu = (1 + d)/2 + \kappa + \nu$ via a global setting or as an optional argument in `cov.args`. This in essence facilitates the use of the covariance function in maximum likelihood estimation.

3.2 Fixing parameters during maximum likelihood estimation

All of the covariance functions contained in the **spam** package include both a range and nugget parameter, which are typically estimated jointly with the shape parameters and partial sill. These additional parameters greatly increase computation time during estimation, and may not be relevant to every user. For these cases, this package offers a wrapper for covariance functions which permits users to fix the range and nugget parameter to a constant value.

Table 1: Simple timing (in ms) of different computation methods.

	Minimum	Q1	Mean	Median	Q3	Maximum
Exact	68.29	68.64	69.09	68.75	69.13	82.71
Askey	0.10	0.12	0.25	0.13	0.14	14.40
Linear Interpolation	1.84	1.90	1.97	1.93	1.97	14.90
Cubic Spline Interpolation	1.85	1.91	2.01	1.94	1.97	19.65
Polynomial Interpolation	1.97	2.02	2.13	2.05	2.09	16.83

3.3 Working with approximations

Given that the purpose of using the generalized Wendland covariance is to alleviate the scalability issue, the cost incurred by numerical integration should not be ignored, as it can become quite substantial for larger data sets. Consider for example a data set with 100 measurements. Assuming that the distance matrix is dense, at worst this would require 4950 evaluations of the correlation function. Approximations can help address this, and as a consequence of correlation functions being positive-definite, they are straightforward to approximate using interpolation. This package provides linear interpolation, cubic spline interpolation, and polynomial interpolation. Table 1 presents timing statistics for one complete evaluation with $n = 100,\ \kappa = 1.25\ \mu = 0.75$

Table 1 provides an overview of computation time across different methods for evaluating the Askey covariance. The comparison between the actual Askey function and the corresponding exact generalized Wendland function illustrates how expensive the latter is to evaluate. For a one-time evaluation this might still be negligible, but not necessarily in maximum likelihood estimation, which generally requires the covariance matrix to be computed in each iteration. Using interpolation, on the other hand, greatly decreases the computational cost of such an evaluation. As Figures 2 and 3 indicate, this is actually a trade-off against the accuracy of the covariance matrix, and results may vary to different extents depending on the choice of interpolator and number of support points. Of particular note in Figure 2 is the erratic nature of the polynomial interpolator for large number of support points. In some applications, it can clearly outperform the other interpolators, yielding absolute errors well below the single precision threshold, yet the interpolated values may also explode. Cubic splines are perhaps the most forgiving method, as the results are barely below the single precision threshold for moderately many support points ($n = 80$).

As curves are visually indistinguishable, but the timing information indicates just how much time was actually saved. A more in-depth analysis of interpolation error is shown in Section 4.

3.4 Compatibility with spam

The **GeneralizedWendland** package was designed for use with the **spam** package, and thus follows its naming convention barring one exception: due to the vast number of optional configuration parameters, these are passed to the function in a list. To ensure that this follows a certain logic, the `eps` argument has also been moved to this list. The function `cov.wendland()` is itself fully compatible with all **spam** functions, as are the wrapper functions generated using `covarianceFactory()`. Furthermore, the package also provides its own suite of tools for parameter estimation, albeit these still require spam as a dependency.

Despite the focus on the spam package, the functions could also be adapted for use in other packages for

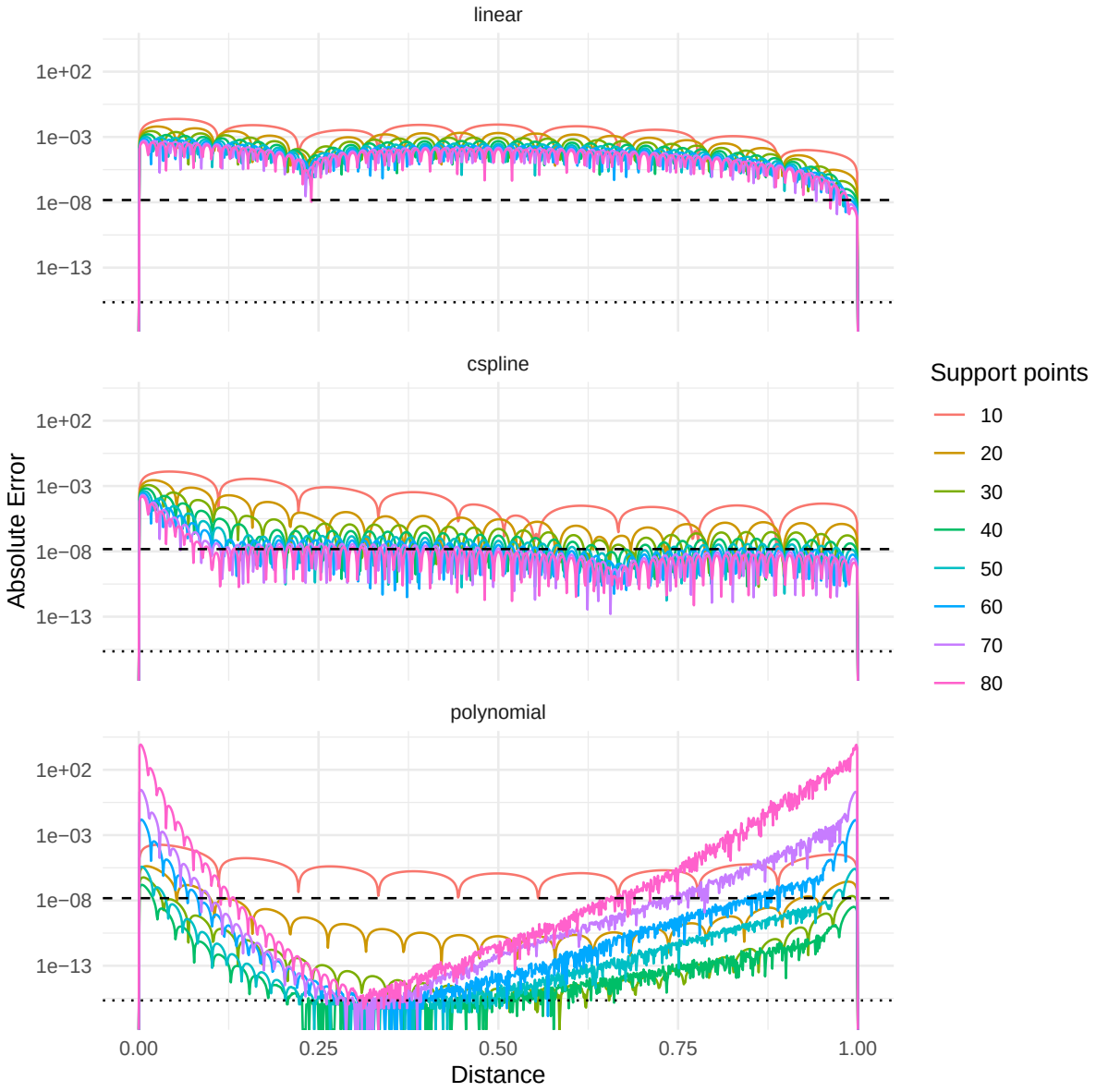


Figure 2: Absolute error of interpolated Wendland correlation function relative to exact method, using $\kappa = 1.25,\ \mu = 3.5$.

geostatistical modeling like the **fields** package. Compatibility to these packages and perhaps to different formats of sparse matrices may be extended in future versions of the **GeneralizedWendland** package.

The package also provides an alternative framework for maximum likelihood estimation which, while based on the **spam** implementation, makes heavy use of function factories to assist the user in setting up their analysis. Finally, the package provides a diagnostic suite which allows users to obtain a selection of error metrics for the generalized Wendland covariance.

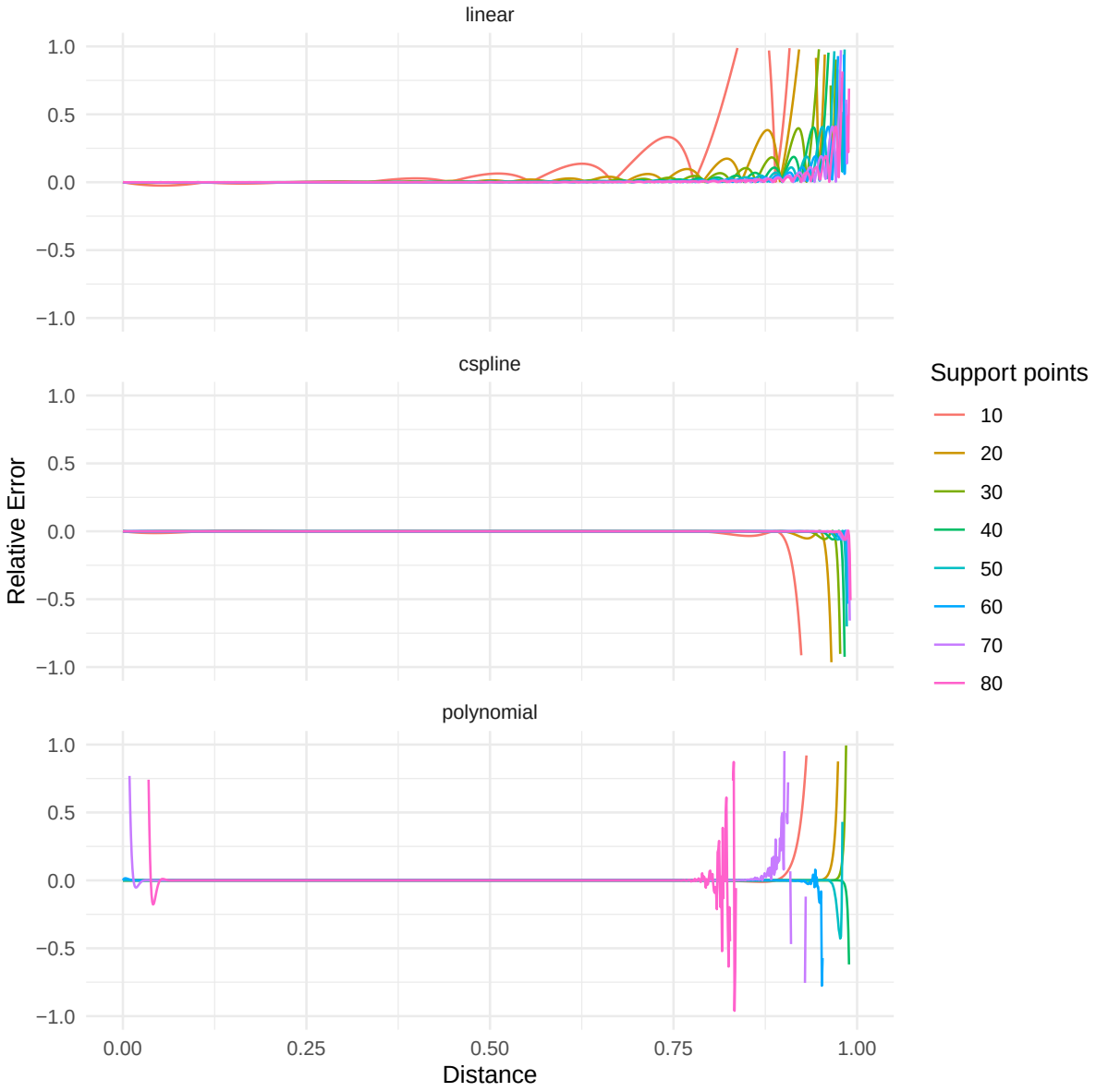


Figure 3: Relative error of interpolated Wendland correlation function compared to exact method, using $\kappa = 1.25,\ \mu = 0.75$.

3.5 Compatibility with optimParallel

When performing maximum likelihood estimation using gradient-based methods such as L-BFGS-B, the gradient needs to be estimated numerically unless the user specifies a gradient function. These operations can be performed in parallel using optimParallel. The mle framework provided in this package is directly compatible with optimParallel, allowing users to specify the relevant arguments in their initial call to the mle function.

4 Technical details

4.1 Implementation of generalized Wendland covariance function

The integral is evaluated via the numerical integration methods provided by the GSL C library. The covariance function itself is implemented as a C++ class and made accessible in R through **Rcpp** (Eddelbuettel, 2013).

4.2 Integration methods

The three available options are

- non-adaptive Gauss-Kronrod integration (QNG, default),
- adaptive integration (QAG),
- adaptive integration with singularities (QAGS).

Figure 4 provides a visual comparison of the results obtained using QNG, QAG, and QAGS integration in the upper left panel, plotting the absolute error in comparison to the Askey function. As immediately apparent, all integration configurations perform well within double precision for whole numbered μ. For rational valued μ, on the other hand, the absolute error just falls short of the single precision threshold, and even slightly exceeds it when using QNG integration. Overall, QAG integration with a high key value performed best among all configurations.

4.3 Covariance interpolation

The three options currently implemented are

- linear interpolation,
- polynomial interpolation,
- Cubic spline interpolation.

These tend to behave very differently with respect to the choice of the number of support points k. Figure 4 shows the absolute error of the interpolated covariance functions compared to the Askey function.

The benchmark against the Askey function shows that polynomial interpolation can potentially outperform the other two using fewer support points, but lacks stability. It tends to explode if using too many support points, and only behaves consistently for whole numbered μ, in which case its absolute error does not exceed the single precision threshold. Linear interpolation, on the other hand, performs very consistently, even for real valued μ. The downside is that it requires a substantial number of support points to obtain approximations with single precision absolute errors. To put this into perspective, linear interpolation would still be worthwhile when working with large data sets, especially since it is the most robust method with regards to μ. Finally, cubic spline interpolation achieves absolute errors below single precision threshold with moderately many support points, and below double precision using a large number of support points.

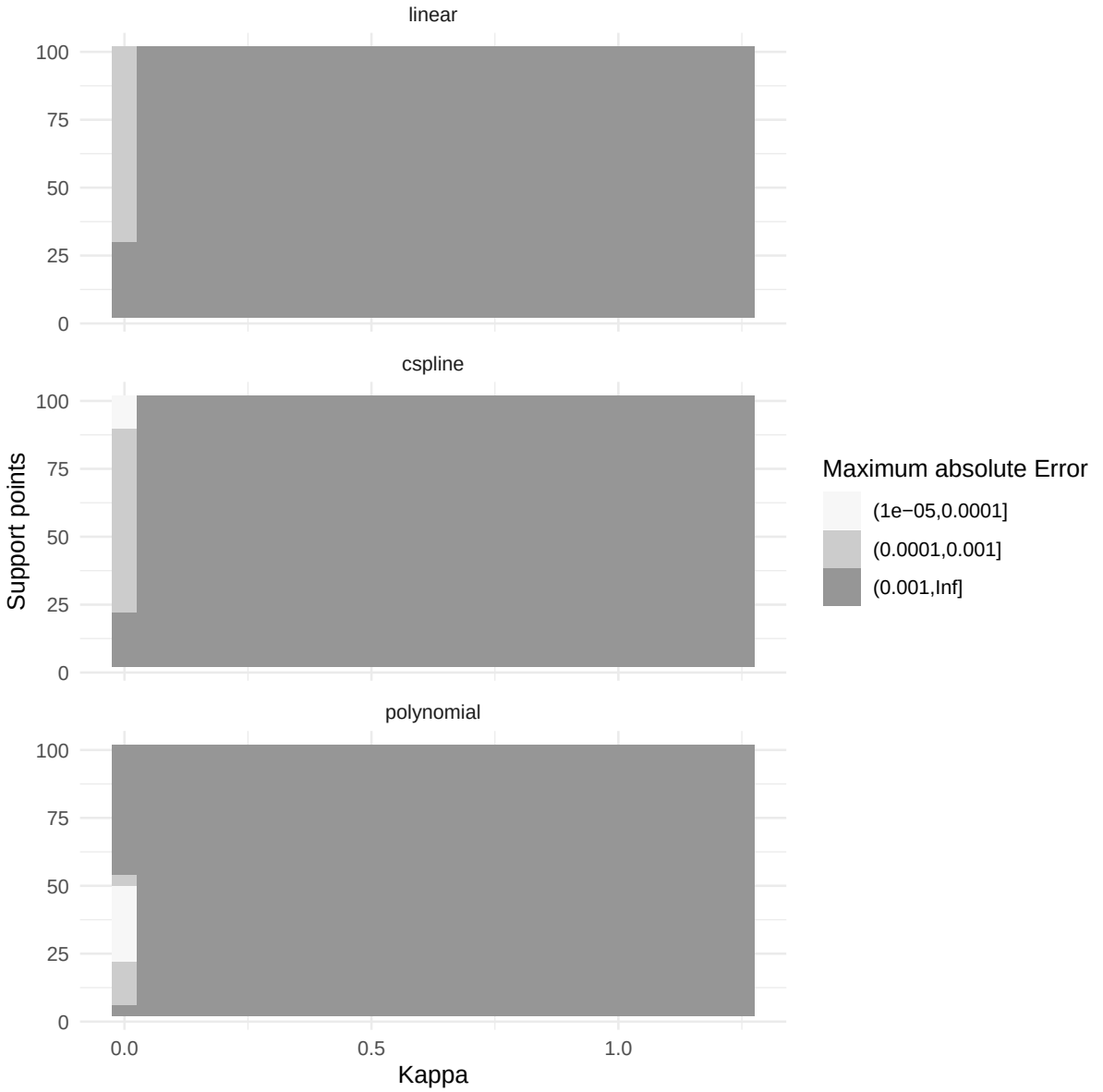
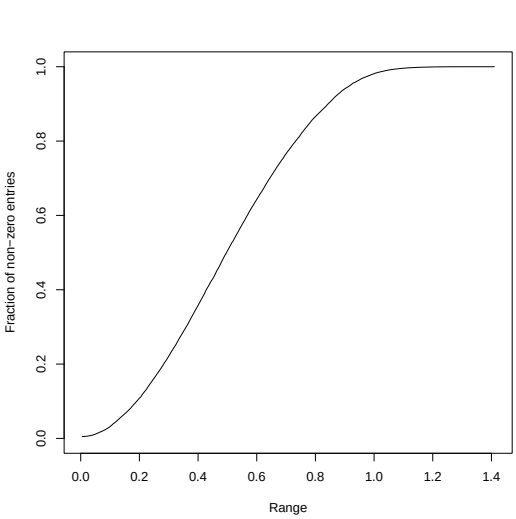


Figure 4: Absolute error of interpolation methods relative to Askey function. Dashed line corresponds to single precision and dotted line to double precision. y-axis is $\log_{10}$ scaled.

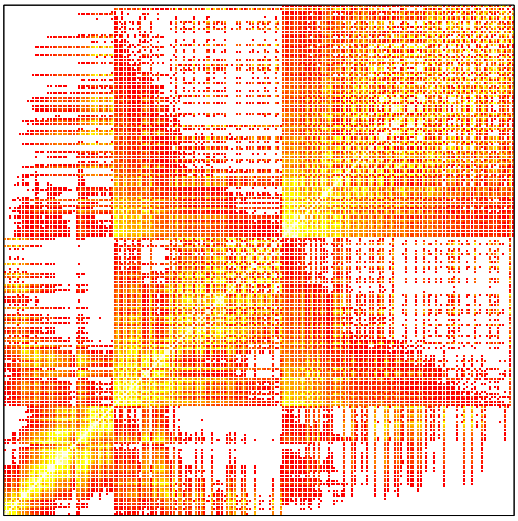
One thing users should keep in mind is that the results shown in Figure 4 also depend on the choice of the numeric tolerances. For example, tuning these parameters can help attenuate the instability of polynomial interpolation, at least to an extent. Users may want to explore feasible settings using the `cov.wendland.diagnostics()` function. For applications such as maximum likelihood estimation, on the other hand, the author recommends choosing either linear or cubic spline interpolation with a reasonably large number of support points.

4.4 Direct misspecification

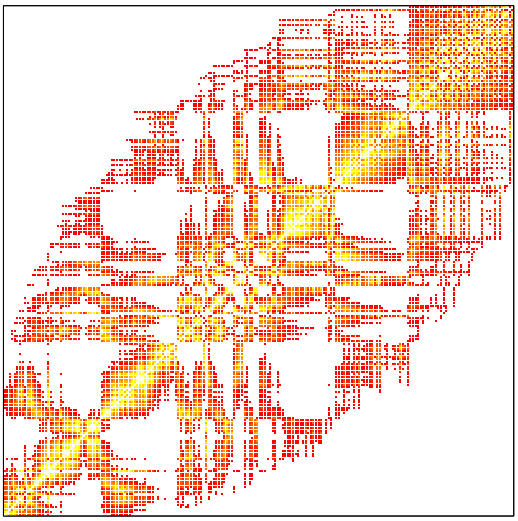
The range parameter is typically estimated jointly with the other covariance parameters. Direct misspecification is the deliberate fixing of the range at an arbitrary value. Note that this will effectively



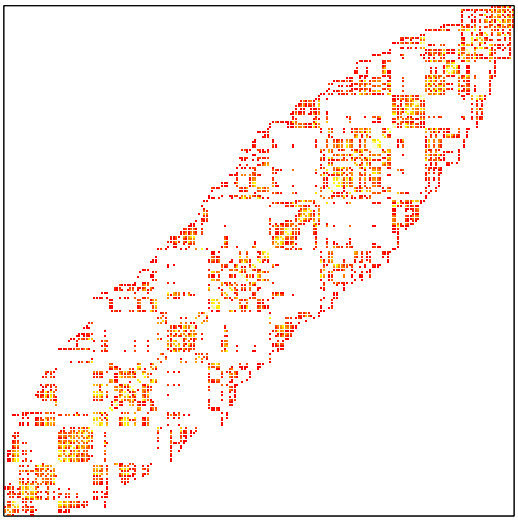
(a) Range and sparsity.



(b) $\beta = 0.6$



(c) $\beta = 0.4$



(d) $\beta = 0.2$

Figure 5: Functional relationship between range and sparsity, and permutated covariance matrices obtained using range $\beta = \{0.6,\ 0.4,\ 0.2\}$.

bias the estimates at least with respect to the range parameter. The motivation for this compromise is twofold: firstly, fixing the range removes one parameter to estimate, resulting in a substantial reduction of computation time. Secondly, reducing the range also reduces the number of non-zero entries in the covariance matrix Σ, which translates into an additional reduction in computation time. The effect of varying ranges on the sparsity of a given covariance matrix is illustrated in Figure 5a, where sparsity is described as a function of range, and in Figure 5 which presents the actual covariance matrices at specific ranges.

To make use of this option for arbitrary covariance functions, users can set the `fixed_range_value` argument in `covarianceFactory`. This will be further illustrated in the next section.

5 Estimating Kriging models using GeneralizedWendland

```
# Simulated spatial data
n <- 1000
grid_resolution <- 2.5e-3#3.33e-3
pred_resolution <- 5e-2

locs <- expand.grid(x = seq(-1, 1, grid_resolution),
                    y = seq(-1, 1, grid_resolution))
max_dist <- sqrt((max(locs$x)-min(locs$x))^2 + (max(locs$y)-min(locs$y))^2)
set.seed(random_seed)

obs_ind <- sample(1:nrow(locs), n)
locs0 <- locs[obs_ind,]
locs1 <- expand.grid(x = seq(-1 + pred_resolution/2,
                             1 - pred_resolution/2,
                             pred_resolution),
                     y = seq(-1 + pred_resolution/2,
                             1 - pred_resolution/2,
                             pred_resolution))

# Specify true model
drift_formula <- ~1 + x + y
true_beta <- c(0.5, -0.1, 0.1)
true_theta <- c(0.7, 3.0, 0.5, 1.5, 1)

# Initial values and box constraints
init_theta <- c(0.7, 1.0, 0.5, 1.5, 0.1)
lower_theta <- rep(0, 5)
upper_theta <- c(max_dist, rep(1e1, 4))

# Design matrices
X0 <- model.matrix(drift_formula, locs0)
X1 <- model.matrix(drift_formula, locs1)

# Distance matrix
dmat <- spam::nearest.dist(locs0, locs0, delta = true_theta[1])

# True covariance matrix
true_Sigma <- cov.wendland(dmat, theta = true_theta)

# Simulate from multivariate normal
y <- c(spam::rmvnorm.spam(1, mu = X0 %*% true_beta, Sigma = true_Sigma))
```

To illustrate how to use **GeneralizedWendland** for geostatistical modeling, this section presents a small, artificial example. For simplicity, we assume that measurement locations occupy an equidistant 2D grid with n =1000 points. Furthermore, the data model includes a global mean and linear spatial drift. As the spatial Gaussian process model is frequently applied to interpolating between measurement

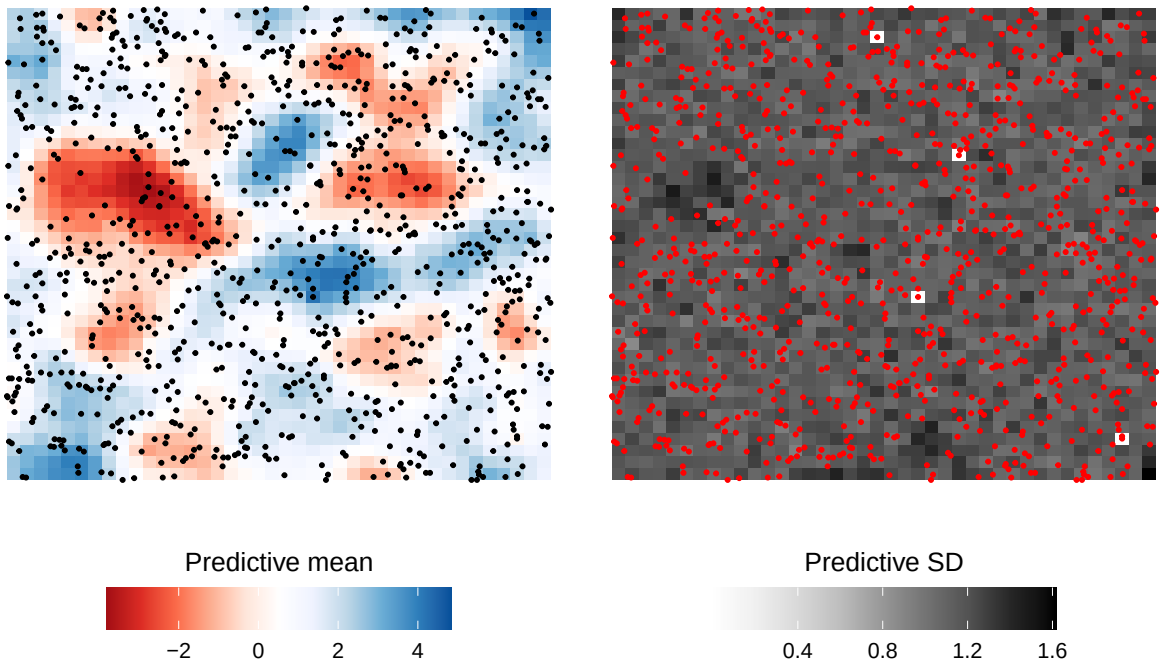


Figure 6: Empiric predictive mean and standard deviation at new locations for true parameters.

locations, Figures 6, 7, 8, 9, and 10 illustrate this by drawing realizations from a multivariate Gaussian distribution and computing the mean and standard deviation per location.

5.1 Exact method

```
cov.args1 <- cov.args <- list()
mleFun <- mleFactory(covariance = cov.wendland, cov.args = cov.args,
  chol.args = chol.args, optim.args = optim.args, hessian = FALSE,
  optimParallel.args = optimParallel.args)

time1 <- system.time({
  result1 <- mleFun(y = y, X = X0, distmat = dmat,
    init_parameters = init_theta, theta_llim = lower_theta,
    theta_ulim = upper_theta)
})

print(result1[c("par", "value", "counts")], digits = digits)


## $par
## [1] 1.04 0.34 0.13 0.59 2.59 0.66 0.91 1.09
##
## $value
## [1] 3342
##
## $counts
```

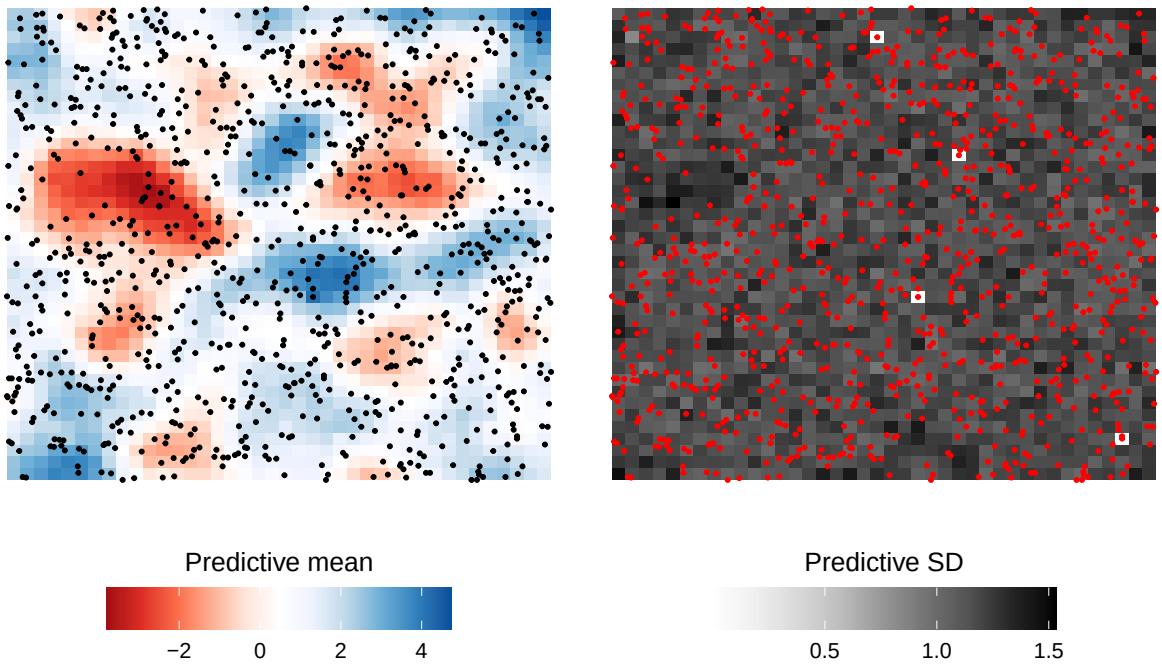


Figure 7: Empiric predictive mean and standard deviation at new locations for parameters estimated using method 1.

```
## function gradient
##       51       51
```

In the basic use case, maximum likelihood estimation utilizes a covariance matrix obtained from the exact formulation of the generalized Wendland function, and all model parameters are estimated jointly. The implementation is mostly equivalent to that provided in the **spam** package.

5.2 Using covariance interpolation

```
cov.args2 <- cov.args <- list(interp.num_support = 100, interp.method = "cspline")

mleFun <- mleFactory(covariance = cov.wendland, cov.args = cov.args,
  chol.args = chol.args, optim.args = optim.args, hessian = FALSE,
  optimParallel.args = optimParallel.args)

time2 <- system.time({
  result2 <- mleFun(y = y, X = X0, distmat = dmat,
    init_parameters = init_theta, theta_llim = lower_theta,
    theta_ulim = upper_theta)
})

print(result2[c("par", "value", "counts")], digits = digits)

## $par
```

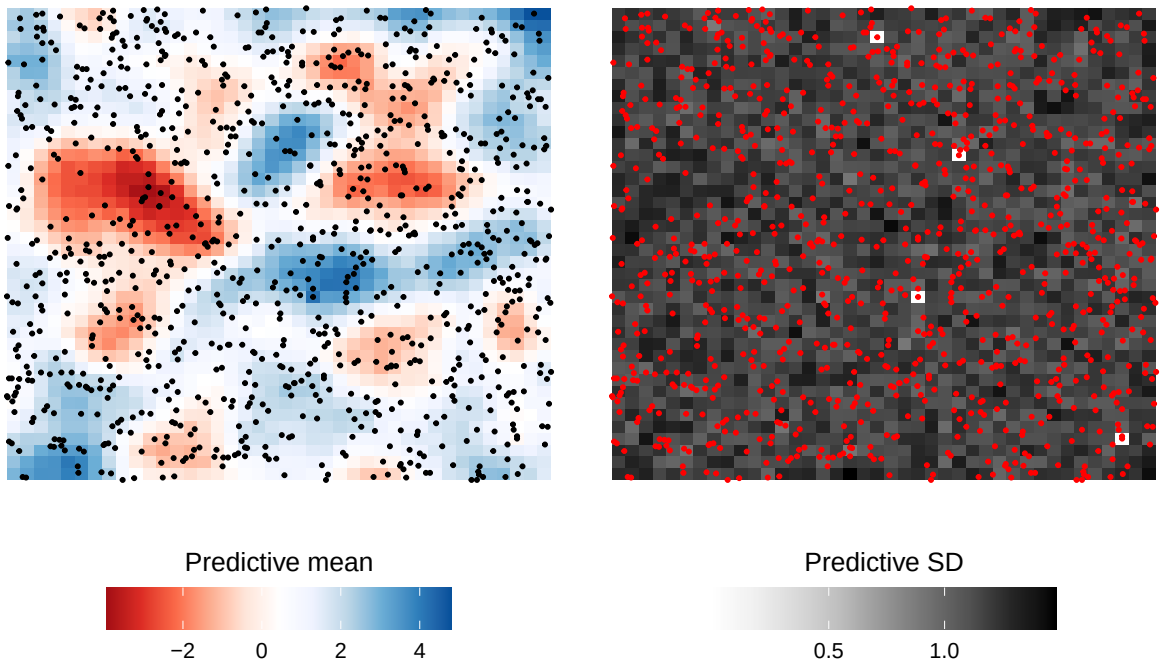


Figure 8: Empiric predictive mean and standard deviation at new locations for parameters estimated using method 2.

```
## [1] 1.04 0.34 0.13 0.59 2.59 0.66 0.91 1.09
##
## $value
## [1] 3342
##
## $counts
## function gradient
##       53       53
```

Covariance interpolation is provided by a C++ wrapper for the GSL library. The R function itself only passes the corresponding parameters to this C++ function, and consequently these methods cannot be accessed directly by the user. To make use of covariance interpolation, users have to provide `interp.method` and `interp.num_support` in `cov.args`.

5.3 Using direct misspecification

```
cov.args3 <- cov.args <- list(fixed_range_value = true_theta[1])

mleFun <- mleFactory(covariance = cov.wendland, cov.args = cov.args,
  chol.args = chol.args, optim.args = optim.args, hessian = FALSE,
  optimParallel.args = optimParallel.args)

time3 <- system.time({
  result3 <- mleFun(y = y, X = X0, distmat = dmat,
```

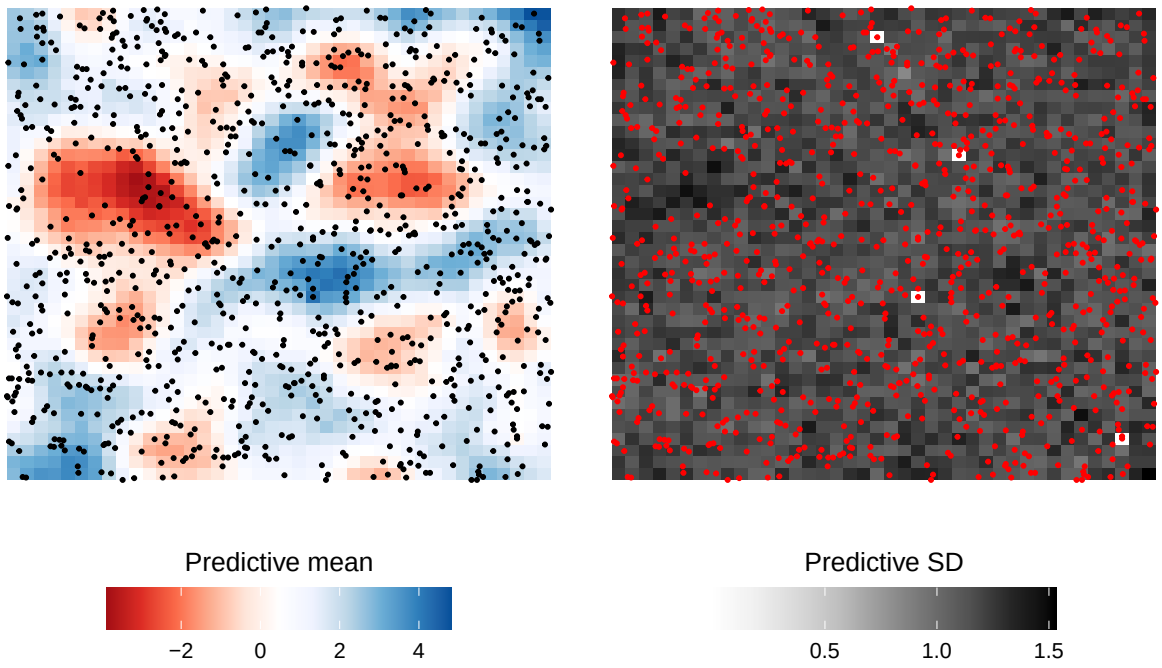


Figure 9: Empiric predictive mean and standard deviation at new locations for parameters estimated using method 3.

```
    init_parameters = init_theta[-1], theta_llim = lower_theta[-1],
    theta_ulim = upper_theta[-1])
})

print(result3[c("par", "value", "counts")], digits = digits)


## $par
## [1] 1.05 0.34 0.14 2.57 0.86 1.79 1.11
##
## $value
## [1] 3343
##
## $counts
## function gradient
##       34       34
```

Direct misspecification is implemented through the `covarianceFactory` function. This is a function factory which essentially serves as a wrapper to any arbitrary covariance function with comparable arguments to `cov.wendland()`. Users can fix the range to a specific value by assigning it to the `fixed_range_value` argument of `cov.args`.

5.4 Combining interpolation and direct misspecification

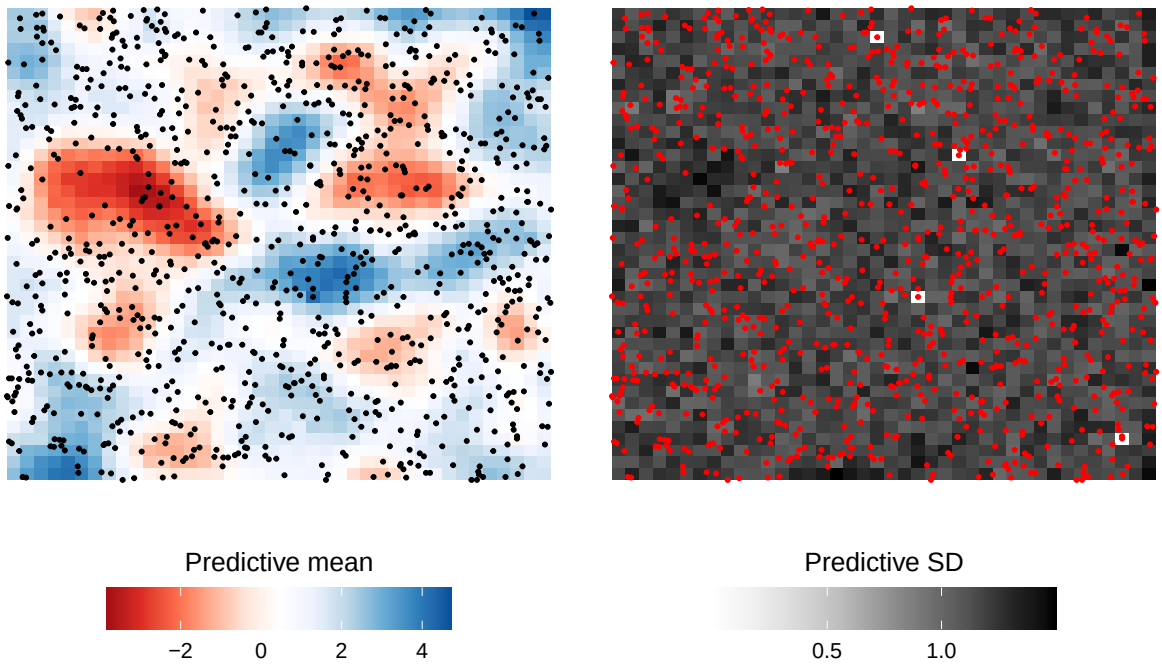
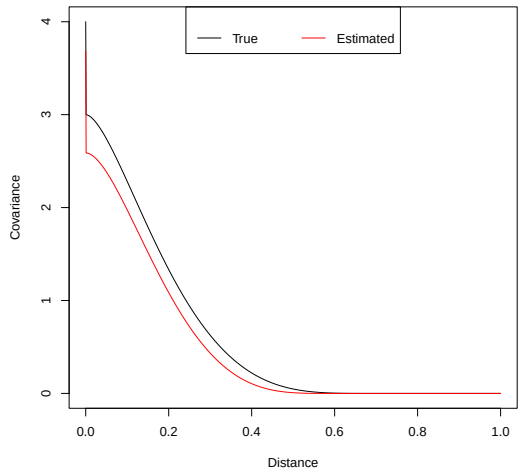
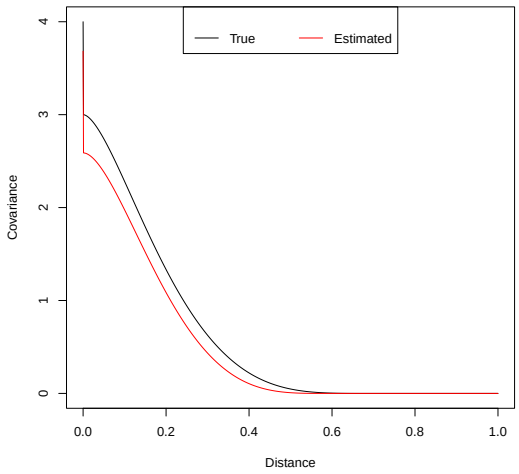


Figure 10: Empiric predictive mean and standard deviation at new locations for parameters estimated using method 4.

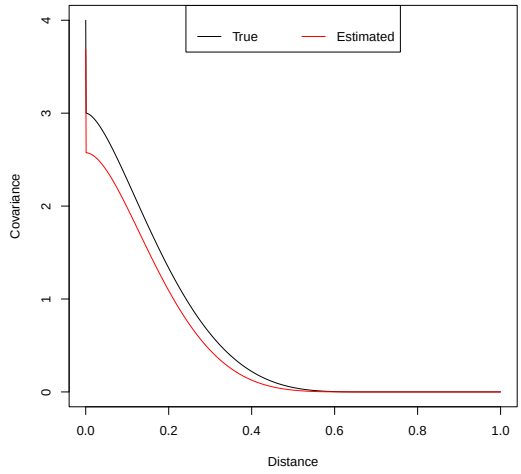
```
cov.args4 <- cov.args <- list(interp.num_support = 100,
  interp.method = "cspline", fixed_range_value = true_theta[1])

mleFun <- mleFactory(covariance = cov.wendland, cov.args = cov.args,
  chol.args = chol.args, optim.args = optim.args, hessian = FALSE,
  optimParallel.args = optimParallel.args)

time4 <- system.time({
  result4 <- mleFun(y = y, X = X0, distmat = dmat,
    init_parameters = init_theta[-1], theta_llim = lower_theta[-1],
    theta_ulim = upper_theta[-1])
})

print(result4[c("par", "value", "counts")], digits = digits)


## $par
## [1] 1.05 0.34 0.14 2.58 0.86 1.79 1.11
##
## $value
## [1] 3343
##
## $counts
## function gradient
##       34       34
```



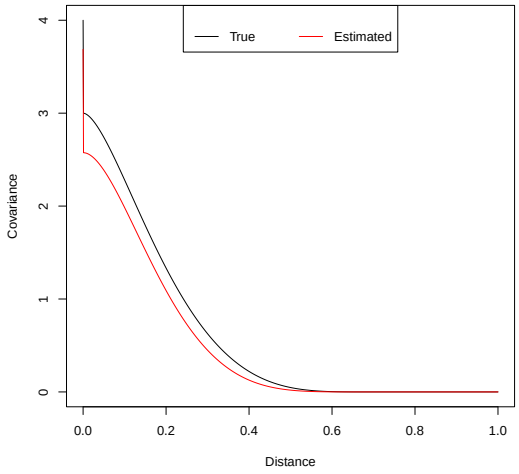
(a) Exact method.



(b) Cubic splines (100 support points).



(c) Direct misspecification ($\beta = 0.7$).



(d) Both (100 support points, β =0.7).

Figure 11: Actual covariance function versus estimated covariance functions.

Direct misspecification and covariance interpolation can also be used conjointly to further reduce computation time.

5.5 Comparison of results

Figure 11 indicates that all methods discussed here yield very similar estimates for the covariance function, the main differences pertaining to the reparameterized μ. The actual estimates are presented in Table 2. Of particular interest are the timings for the maximum likelihood estimation itself. For the exact approach and direct misspecification with a reasonable range, computation time is eclipse 3 minutes, whereas using interpolation reduces this down to just over a minute, and even just 40 seconds when using both interpolation and misspecification.

Table 2: Overview of results and computation time for all examples.

	Intercept	x-drift	y-drift	range	sill	kappa	mu	nugget	elapsed
Actual	0.50	-0.10	0.10	0.70	3.00	0.50	1.50	1.00	NA
Exact	1.04	0.34	0.13	0.59	2.59	0.66	0.91	1.09	249.88
Interpolation	1.04	0.34	0.13	0.59	2.59	0.66	0.91	1.09	66.73
Direct Misspecification	1.05	0.34	0.14	0.70	2.57	0.86	1.79	1.11	217.55
Both	1.05	0.34	0.14	0.70	2.58	0.86	1.79	1.11	42.47

6 Conclusion

The preceding sections provided a brief overview of the generalized Wendland covariance function and its properties, and illustrated several of the features which are present within the **GeneralizedWendland** package. The most important features of this package are the accessible implementation of the generalized Wendland covariance, a framework for direct misspecification of the range parameter, and covariance interpolation. All of these features are of particular usefulness for geostatisticians working with large datasets, as for example obtained from remote sensing. Such applications may benefit from adjustable sparsity.

References

Bevilacqua, M., Caamaño-Carrillo, C., and Porcu, E. (2022). Unifying compactly supported and matern covariance functions in spatial statistics. arXiv:2008.02904v3.

Bevilacqua, M., Furrer, R., Faouzi, T., and Porcu, E. (2019). Estimation and prediction using generalized wendland covariance functions under fixed domain asymptotics. *The Annals of Statistics*, 47 828–856.

CRAN (2022). Rtools42 for windows. `https://cran.r-project.org/bin/windows/Rtools/rtools42/rtools.html`.

Eddelbuettel, D. (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York.

Eddelbuettel, D. and Francois, R. (2022). *RcppGSL: 'Rcpp' Integration for 'GNU GSL' Vectors and Matrices*. R package version 0.3.11, `https://CRAN.R-project.org/package=RcppGSL`.

Furrer, R., Flury, R., and Gerber, F. (2022). *spam: SPArse Matrix*. R package version 2.8-0, `https://CRAN.R-project.org/package=spam`.

Galassi, M., Davies, J., Theiler, J., and Gough, B. (2021). *GNU scientific library reference manual (2.7)*. `https://www.gnu.org/software/gsl/doc/latex/gsl-ref.pdf`.

Gerber, F. and Furrer, R. (2019). optimParallel: An R Package Providing a Parallel Version of the L-BFGS-B Optimization Method. *The R Journal*, 11 352–358.