Quasi-likelihood Estimation with R

Markus Baaske

Abstract

We introduce the R package **qle** for simulation-based quasi-likelihood parameter estimation. We briefly summarise the basic theory of quasi-likelihood for our setting and outline the algorithmic framework of the proposed method. We focus on the general workflow using the package and present two introductory examples. Finally, we apply the method to an example data set from spatial statistics.

Keywords: quasi-likelihood, simulation-based estimation, black box optimization, kriging metamodel, cross-validation, uncertainty, R.

1. Introduction

The R package **qle** implements methods for parametric inference for a generic class of estimation problems where we can at least simulate a (hypothesized) statistical model and compute certain summary statistics of the resulting model outcome. The method is an extended and modified version of the algorithm proposed in **?**. The aim of this paper is to present the algorithmic framework of our simulation-based quasi-likelihood estimation (SQLE) approach for parametric statistical models. The method employes the first two moments of the involved summary statistics by simulation and constructs a quasi-likelihood approximation for estimating the statistical model parameter. An application of our method to the problem of inferring parameters of a spheroid distribution from planar sections can be found in **?**.

We assume that, given the data, no closed-form expressions or direct computation algorithms of likelihood functions or distribution characteristics, as functions of the model parameter, are available. One may also be unable or reluctant to formulate a likelihood function for complex statistical models. This precludes many standard estimation methods, such as *maximum likelihood* (ML) estimation, typical Bayesian algorithms (including Markov-Chain-Monte-Carlo-type algorithms), or (generalized) least squares (LS) based on exact distribution characteristics. In many settings, it is still conceptually possible to consider some observed data as a realization of a vector of random variables whose joint distribution depends on an unknown parameter. If a statistical model provides enough information about the data generating process, we can think of it as partially specifying some aspects of the distribution. However, it may be still possible to approximately compute characteristics of the data as Monte Carlo estimates from computer simulations. Such characteristics can be any sort of descriptive summary statistics which carry "enough" information about the unknown dependency between simulated realizations and the model parameter. Usually, these "informative" statistics are context-dependent and can be specifically tailored for the setting of the parameter estimation problem under investigation.

1.1. Quasi-likelihood approach

If we could simulate the statistical model fast enough such that the Monte Carlo error of the statistics becomes neglectable, a general approach for estimating the model parameter would be to minimize some measure of discrepancy, i. e. a criterion function, finding those parameters which lead to simulated statistics most similar to the observed ones in the sense of a least squares, minimum contrast criterion, or, more generally, estimating functions approach, see **?** and also **?**[1]. The estimator of the true model parameter could then be found by a standard numerical solver. However, complex statistical models usually require very time-consuming, computationally intensive simulations and typically the Monte Carlo error cannot be made small enough within a reasonable timeframe such that any direct application of the above estimation approaches would become numerically infeasible.

Conceptually, our method is based on the class of linear estimating functions in its most general form of minimum (asymptotic) variance unbiased estimation. We adapt the so-called *quasi-score* (QS) estimating function to the setting where generated realizations of the statistical model can be characterised by a set of appropriately chosen summary statistics. The derivation of the QS is part of the general approach of *quasi-likelihood* (QL) estimation, see **?**, which subsumes standard parameter estimation methods such as, for example, ML or (weighted) LS. The QL estimator is finally derived from the solution to the QS equation (see Section 2). As a common starting point the QS can be seen as a gradient specification similar to the score vector in ML theory. If a likelihood is available, both functions coincide and the score function from ML is an optimal estimating function in the sense of QL.

Except in some rare cases, when expectations, derivatives thereof and variances of the statistics are known at least numerically, any kind of criterion function derived from one of the above criteria, including the QL approach, would lack a closed-form expression and could only be computed slowly with substantial random error either due to the inherent simulation variance or erroneous evaluation of the involved statistics. In fact, nothing is said about a QL function in theory which could be employed as an objective function for minimization in order to derive an estimate of the true model parameter. Therefore, our idea is to treat such a function as a black box objective function and to transform the general parameter estimation problem into a simulation-based optimization setting with an expensive objective function. For this kind of optimization problem it is assumed that derivative information is either not available or computationally prohibitive such that gradient-based or Newton-type methods are not directly applicable.

1.2. Background on black box optimization

A general problem which holds both for finding a minimum of an expensive objective function or a solution to the QS equation is to efficiently explore the parameter search space when only a limited computational budget is available for simulation. A suitable approach for this kind of problem relies on the use of response surface models in place of the original black box function for optimization. Examples include first-degree and second-degree (multivariate) polynomial approximations of response functions, which are broadly used, for example, in the *response surface methodology*, see **?**. Another approach includes the *kriging methodology* (see, e. g. **???**) which treats the response of an objective function as a realization of a stochastic process. The main idea is to start by evaluating the expensive function at sampled points

[1] For a more recent review on estimating functions and its relation to other methods

of a generated (space-filling) experimental design over the whole parameter space. Then a global response surface model is constructed and fitted based on the initially evaluated design points and further used to identify promising points for subsequent function evaluations. This process is repeated, now including the newly evaluated points, for sequential updates of the response surface model. In order to improve the model within subsequent iterations the aim is to select new evaluation points which help to estimate the location of the optimum, that is, the unknown parameter in our setting, and, at the same time, identifying sparsely sampled regions of the parameter search space where little information about the criterion function is available.

In this context, kriging has become very popular mainly for two reasons: first, it allows to capture and exploit the data-inherent smoothness properties by specifically tuned covariance models which measure the spatial dependency of the response variable and, second, it provides an indication of the overall achievable prediction or estimation accuracy. Almost all kriging-based global optimization methods, such as the well-known *Efficient Global Optimization* algorithm by **?**, are based on the evaluation of kriging prediction variances in one way or another[2]. Although these models have been widely used in the community of global optimization of expensive black box functions with applications to engeneering and economic sciences these seem to be less popular in the context of simulation estimation.

1.3. Main contribution

Opposed to the general framework of black box optimization, where some scalar-valued objective function is directly constructed via kriging, we propose the use of kriging models for each involved summary statistic separately because, unlike the QS function itself, only the statistics can be estimated unbiasedly from simulations. Based on these multiple kriging models we construct an approximating QS estimating function and estimate the unknown model parameter as a root of the resulting QS vector. Therefore, our main contribution is to combine the QL estimation approach with a black box framework that allows to handle time-consuming simulations of complex statistical models for an efficient estimation of the parameter of a hypothesized true model when only a limited computational budget is available. Besides this, the use of kriging models enables the estimation procedure to be guided towards regions of the parameter space where the unknown model parameter can be found with some probability and hence helps to save computing resources during estimation.

It is worth noting that there exist other R packages which make use of the QL concept in one or another way. For example, the function `glm` (**?**) for fitting a *generalized linear model* is closely related to the estimation theory by **?** which is known under the general term of *quasi-likelihood methods*. From the viewpoint of estimating functions this approach can be considered a special case of the more general QL theory in **?**. Also the package `gee` (**?**) is made for a rather different setting which uses the moment-based *generalized estimating equations*, see **?**. This package mostly deals with analysing clustered and longitudinal data which typically consist of a series of repeated measurements of usually correlated response variables. Further, the package **gmm** (**?**) implements the *generalized method of moments* (**?**) for situations in which the moment conditions are available as closed-form expressions. However, if the population moments are too difficult to compute, one can apply the *simulated method of moments* (SMM), see **?**. The moment conditions are then evaluated as functions of

[2]See **?** for a comprehensive overview of kriging-based surrogate modelling.

the parameter by Monte Carlo simulations. Also, the package **spatstat** (**?**) includes a so-called quasi-likelihood method. However, the implementation is specifically made for the analysis of point process models.

Our method is specifically designed to deal with situations where only a single measurement (i. e. "real-world" observation or raw data) is available and from which we can compute the (observed) summary statistics. To this end, we assume that we can define and simulate a parametric statistical model which reveals some information about the underlying data generating process. Then the parameter of interest (under this model) is inferred from a solution to the QS equation based on these statistics. The computational complexity of our method is, therefore, mostly dominated by the effort of simulating the statistical model and evaluating the involved statistics. The main aim of the package is to provide an efficient implementation for generic parameter estimation problems which combines the involved statistics into a tractable estimating function for simulation when no other types of parametric inference can be applied.

The vignette is organized as follows. In Section 2 we briefly present the basic theory of QL estimation followed by an outline of the main algorithm in Section 3. We discuss some extensions of our originally proposed algorithm in Section 4. Finally, in Section 5, we provide some illustrative examples of a typical workflow of using the package.

2. Basic quasi-likelihood theory

In this section we sketch the main aspects of the general QL theory, see **?**. Let X be a random variable on the sample space $\mathcal{X}$ whose distribution P_θ depends on the unknown parameter $\theta \in \Theta$ taking values in an open subset Θ of the q-dimensional Euclidean space $\mathbb{R}^q$. The goal is to estimate θ given the observed data $x = X$. We assume that the family of models P_θ is rich enough to characterise and distinguish different values of the generative parameters. Let $T : \mathcal{X} \to \mathbb{R}^p$ with $p \geq q$ be a transformation function of the data to a set of summary statistics and $y = T(x)$ is the respective (column) vector of summary statistics. We follow the QS estimating function approach to estimate θ by equating the QS

$$Q(\theta, y) = \left(\frac{\partial \mathsf{E}_\theta[T(X)]}{\partial \theta}\right)^t \mathrm{Var}_\theta(T(X))^{-1}(y - \mathsf{E}_\theta[T(X]) \tag{2.1}$$

to zero, where $(\cdot)^t$ denotes transpose, and, respectively, E_θ and Var_θ denote expectation and variance w.r.t. P_θ.

For a fixed vector of summary statistics $T \in \mathbb{R}^p$ we focus on the function $G(\theta, y) = y - \mathsf{E}_\theta\left[T(X)\right]$ as a vector of dimension p, for which $\mathsf{E}_\theta\left[G\right] = 0$ for each P_θ and for which the matrices $\mathsf{E}_\theta\left[\partial G/\partial\theta\right]$ and $\mathsf{E}_\theta\left[GG^t\right]$ are nonsingular. The QS estimating function in (2.1) is the standardized estimating function

$$\tilde{G} = -\left(\mathsf{E}_\theta\left[\frac{\partial G}{\partial \theta}\right]\right)^t (\mathsf{E}_\theta\left[GG^t\right])^{-1} G \tag{2.2}$$

for which the information criterion

$$\mathcal{E}(G) = \mathsf{E}_\theta\left[\tilde{G}\tilde{G}^t\right] = \left(\mathsf{E}_\theta\left[\frac{\partial G}{\partial \theta}\right]\right)^t (\mathsf{E}_\theta[GG^t])^{-1} \left(\mathsf{E}_\theta\left[\frac{\partial G}{\partial \theta}\right]\right) \tag{2.3}$$

is maximized in the partial order of non-negative definite matrices among all *linear* unbiased estimating functions of the form $A(\theta)(y - \mathsf{E}_\theta\,[(T(X)]$, where $A(\theta)$ is any nonsingular matrix. The information criterion in (2.3) is seen as generalization of the well-known Fisher information since it coincides with the Fisher information in case a likelihood is available such that G equals the score function in ML theory. Then, in analogy to ML estimation, the inverse of $\mathcal{E}(G)$ has a direct interpretation as the asymptotic variance of the estimator $\hat{\theta}_0$. Moreover, $\mathcal{E}(G)$ might serve as a measure of how much the statistics T contribute to the precision of the estimator derived from the QS equation. Under rather minor regularity conditions, the estimator $\hat{\theta}_0$ obtained from solving the estimating equation $Q(\hat{\theta}_0, y) = 0$ has minimum asymptotic variance among all functions G and consistency (see, e. g. **?**) is established due to the unbiasedness assumption of the estimating equation in (2.1) which yields, in terms of its root, a consistent estimator $\hat{\theta}_0$ even if the covariance structure is not correctly specified. The information criterion in (2.3), given a vector T of summary statistics, then reads

$$I(\theta) = \mathsf{Var}_\theta(Q(\theta, T(X))) = \left(\frac{\partial \mathsf{E}_\theta\,[T(X)]}{\partial \theta}\right)^t \mathsf{Var}_\theta(T(X))^{-1} \left(\frac{\partial \mathsf{E}_\theta\,[T(X)]}{\partial \theta}\right), \tag{2.4}$$

which we call *quasi-information* (QI) matrix in what follows. In particular, if we had an analytical or at least a numerically tractable form of the involved expectations and variances, we could apply a gradient based method to solve the QS equation similar to finding a root of the score vector in ML estimation. However, since closed-form representations of expectations and variances are generally unavailable for complex statistical models or prohibitive to compute we assume that we can only simulate realizations of the random variable X under P_θ at any $\theta \in \Theta$.

3. Simulated quasi-likelihood estimation method

Let $Z : \Theta \to \mathbb{R}^p$ be a function of the model parameter $\theta \in \Theta$ to the expected value of T, i. e. $Z(\theta) = \mathsf{E}_\theta[T(X)]$, and let $V(\theta) = \mathsf{Var}_\theta(T(X))$ denote the variance of the statistics under P_θ. Since we assume that we can only infer $T(X)$ by simulated realizations of X we treat Z and V as deterministic black box functions, which could be very expensive to evaluate in practice. In order to compute the QS function at θ the sample means of the statistics and the variance V are approximated by kriging models, denoted as $\hat{Z}$ and, respectively, $\hat{V}$. Since both functions can only be evaluated with a random error due to the simulation replications of the statistical model we explicitly account for the resulting approximation error in the construction of the QS function by adding the kriging prediction variances of the sample means of the statistics (seen as a measure of predictive accuracy) to $\hat{V}$ as diagonal terms (see Section 4.3). Then, the resulting approximation of the QS reads

$$\hat{Q}(\theta, y) = \hat{Z}'(\theta)^t \hat{V}(\theta)^{-1}(y - \hat{Z}(\theta)) \tag{3.1}$$

given the observed statistics $Y = y$. Analogously, the approximation of the information matrix in (2.4) is

$$\hat{I}(\theta) = \hat{Z}'(\theta)^t \hat{V}(\theta)^{-1} \hat{Z}'(\theta), \tag{3.2}$$

where $\hat{Z}' \in \mathbb{R}^{q \times p}$ denotes the Jacobian (the matrix of partial derivatives of $\hat{Z}$) which is numerically computed by finite differences based on the kriging approximations of Z. The included diagonal terms play the role of individual weights for the vector of contrasts $y - \hat{Z}(\theta)$

in (3.1) according to the predicted error of $\hat{Z}$. Likewise, for the QI matrix in (3.2), the partial derivatives of the kriging predictor $\hat{Z}$ are similarly weighted. Thus, the above QS accounts for the noises due to the simulation replications of the statistical model. Note that, compared to the typically high simulation effort required to obtain an estimate of the statistics, the approximations of the QS function and QI matrix can be computed inexpensively once the kriging models have been constructed. Besides the kriging approach to approximate V the package additionally provides other types of variance average approximations, see **?**, including a kernel-weighted verion based on the QI matrix.

Further, the SQLE method implements two criterion functions for estimation. The first one is directly derived from quasi-likelihood theory. The so-called *quasi-deviance* (QD) is defined by

$$s(\theta) = \hat{Q}(\theta)^t \hat{I}(\theta)^{-1} \hat{Q}(\theta), \tag{3.3}$$

where we suppress the dependency on the observed statistics y for convenience. In particular, the QS and QD are considered as inexpensive approximations of their respective counterparts if T could be computed from X under the model P_θ without error at any $\theta \in \mathcal{D}$. The QD - roughly speaking - measures the "deviance" of the QS vector from zero and thus provides a global view on the estimation problem with the goal of solving the QS equation. Opposed to this, the Mahalanobis distance (MD), which reads

$$m(\theta) = \{y - \hat{Z}(\theta)\}^t \, \Sigma_\theta^{-1} \{y - \hat{Z}(\theta)\}, \tag{3.4}$$

where Σ is a positive definite matrix, such as an estimate of the asymptotic variance of y, has a direct interpretation as a (weighted or generalized) least squares criterion depending on the employed type of variance matrix approximation. It can be seen as a special case of SMM, where, in our implementation, either $\Sigma_\theta = \Sigma(\theta)$ is evaluated as a function of θ (using Monte Carlo simulations) or considered as a "constant" estimate for fixed θ. In principle, both criterion functions can be used to guide the estimation procedure towards a (possibly local) minimum or a root of the QS function. However, in contrast to the QD, the gradient of the MD for constant Σ,

$$m'(\theta) = \hat{Z}'(\theta)^t \Sigma^{-1} (y - \hat{Z}(\theta)), \tag{3.5}$$

is readily available (ignoring irrelevant factors) and thus can be used to minimize the criterion function by gradient based methods. This makes the MD also desireable in terms of efficiency and numerical stability for local searches. Obviously, in case the problem dimension equals the number of summary statistics, i. e. $p = q$, both criterion functions coincide. Note that since the MD can be subsumed under the general term of *quasi-likelihood*, different versions (see documentation) of the MD are implemented in the package mainly for two reasons: first, the MD could be used as a preliminary step in order to identify suitable starting points for parameter estimation and, secondly, to assess the goodness-of-fit of the estimated model parameter.

3.1. Algorithmic overview

An outline of the algorithmic framework of our method is given in Figure 1. The basic algorithm starts by generating an experimental design of size n_0 of space-filling points, i. e. the parameters in our setting, and evaluates the statistics at these initial design points by simulation. We say that an *evaluated point* is a point of the search space $\mathcal{D}$ where the statistical

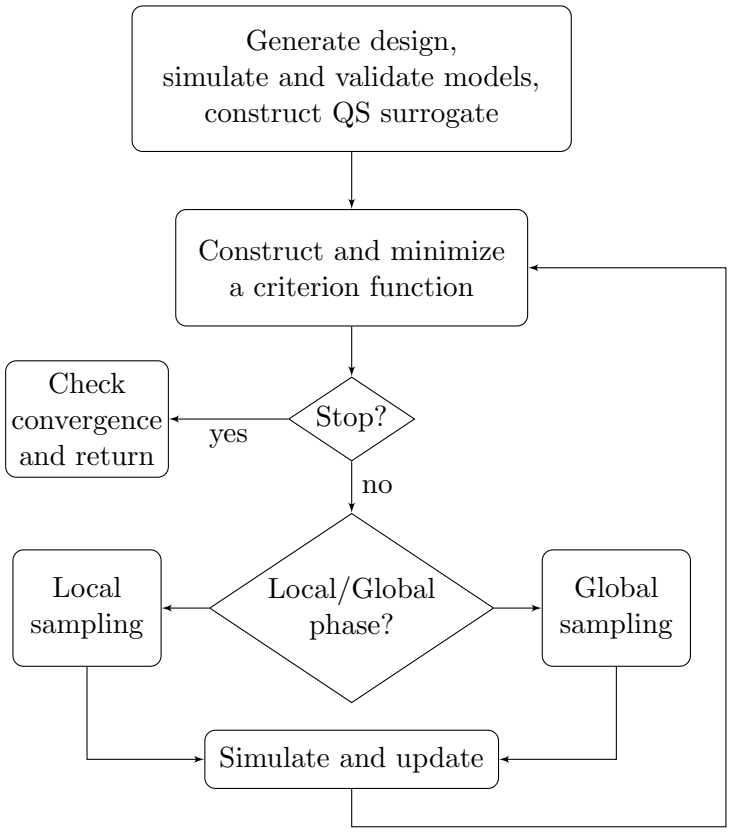

Figure 1: Algorithmic strucure of function `qle`

model is simulated w.r.t. to this parameter or synonymously called "point". Based on the same set of sample points and initially simulated statistics, the kriging models are separately constructed for each statistic $T^{(j)} \in \mathbb{R}$, $j = 1, \ldots, p$ so that the algorithm needs to maintain and update p kriging models simultaneously each time a new evaluation point is added. Also, the user has the option to analyse the initially generated design based on the chosen covariance structure in terms of its adequacy for both the observed and simulated values of the statistics, the prediction accuracy and the basic assumptions of the data phenomenon (see Section 4.5). Moreover, the prediction variance for each point is used to account for the approximation error of the QS vector depending on the current iterate θ each time the algorithm requests a new function value of the QS or the criterion function. In particular, finding a solution to the subproblems in (3.6) and (3.9) also requires to evaluate either of them.

During the main iteration the algorithm updates the QS approximation and the criterion function based on newly evaluated design points, that is, we sequentially select new parameters of the search space for simulating the statistical model by the dual goal of improving the approximations and, at the same time, narrowing the region of possible solutions of the QS function. To achieve this, the algorithm is split into a local and global phase (see Section 4.1 and 4.2) as shown in Figure 1, each using its own sampling and selection criterion for new candidates and search methods to improve the current parameter estimate. In particular, the global phase explores unvisited regions based on one of the criterion functions where in some sense is evidence to suspect a global or local solution to the parameter estimation problem. On

the contrary, the local phase exploits promising local regions for small values of the criterion function in the vicinity of an approximate root or a minimizer. If the value of the criterion function drops below a user-defined tolerance at some point, we immediately switch to the local phase and, otherwise, continue with sampling in the global phase. Thus, the algorithm is allowed to dynamically switch the phases depending on the number of successful iterations towards a potential parameter estimate. The proposed sampling strategies and selection criteria ensure that the search space becomes overall densely sampled if the algorithm is allowed to iterate infinitly often. Moreover, it keeps a ballance between global and local search steps which is essential for the efficiency of the method in practical applications and, at least in theory, guarantees to find a global minimum.

3.2. Approximately solving the quasi-score equation

Througout this section we use the QD as the only criterion function for monitoring the estimation. We aim on estimating $\theta_0 = \theta_0(T)$ for given T as the parameter of the hypothesized true model P_{θ_0}. Using $\hat{Q}$ in (3.1) we deduce an estimate $\hat{\theta} \in \mathcal{D}$ from a solution to the approximating QS equation

$$\hat{Q}(\theta) \approx 0 \tag{3.6}$$

by a *Fisher quasi-scoring iteration* (see, e. g. **?**). In practice, we observed that solving (3.6) might cause numerical instabilities for small- or medium-sized sets of (initial) sample points due to potential approximation inaccuracies of the resulting QS equation. Therefore, even if there are no roots identifiable, we aim on searching for at least an approximate (local) minimizer of one of the criterion functions and then proceed using this last estimate. If we neither assume that equation (3.6) has a unique root nor that it exists, then, a common approach is to restrict the estimation to "plausible" regions of the parameter space as to guarantee the existence of a solution. Therefore, we define the parameter search space by

$$\mathcal{D} = \{\theta \in \Theta \ : \ \theta_l \leq \theta \leq \theta_u\} \subset \Theta, \tag{3.7}$$

where $\theta_l \in \mathbb{R}^q$ and $\theta_u \in \mathbb{R}^q$ denote the lower and upper bounds, respectively, as simple bound constraints. In case of multiple roots, restricting to reasonably chosen smaller subregions might also be a pragmatic solution to distinguish between them. Another approach is given in Section 3.4.

In our implementation, the quasi-scoring algorithm is defined as a projected Newton-type iteration with the update

$$\begin{aligned} h &= \hat{I}(\theta)^{-1}\hat{Q}(\theta), \\ \theta &\leftarrow \mathcal{P}_D[\theta + \alpha h], \end{aligned} \tag{3.8}$$

where $\mathcal{P}_D$ denotes the projection operator on $\mathcal{D}$ and $0 < \alpha \leq 1$ is a step size parameter for the quasi-scoring correction $h \in \mathbb{R}^q$. In contrast to the usual setting of ML estimation the quasi-scoring iteration lacks explicit values of an objective function, monitoring the progress of the iterates towards a root, because of the (in theory) left undefined quasi-likelihood function. A natural way to measure the progress (adjusting the correction h and step length parameter α accordingly) is to use the QD as a monitor function. Therefore, the purpose of the QD is twofold: on the one hand, it is used as a surrogate objective function to monitor the root finding and guarantees the existence of a global minimum over $\mathcal{D} \subset \Theta$. On the other hand, given the observed statistics, we can also set up a hypothesis (see Section 4.7) of the estimated parameter to be the true one based on the approximating QS and QD as a test statistic.

As a surrogate objective function, a global minimizer of the QD takes the QS closest to zero (in the limit of a sequence of surrogate minimizations) and hence is considered to be an estimate, not necessarily unique, of the unknown model parameter. However, the quasi-scoring iteration might fail to converge in practice mainly for two reasons. Besides pathological cases, for example, if no true root exists in $\mathcal{D}$, there are no theoretically backed up tests of some sort of sufficient decrease condition (by contrast to, for example, the Goldstein test or Armijo condition for ML estimation), which would ensure global convergence to a root of the QS by a suitable line search strategy. Therefore, we follow a hybrid search strategy in our implementation as follows. If the quasi-scoring iteration fails, we switch to some *derivative-free optimization* method (see, e. g. **?**) in order to approximately solve the QS equation because we assume that derivatives of any of the involved quantities of the QD are unavailable or computational prohibitive to obtain. In case of non-convergence, we search for a minimizer of the (asymptotically) equivalent optimization problem

$$\min s(\theta) \tag{3.9}$$

$$\text{s.t. } \theta \in \mathcal{D}. \tag{3.10}$$

However, the condition that h is a descent direction, particularly for the QD function, must be also checked to ensure a reasonable descrease after each updated correction. In order to avoid additional expensive numerical computations of function values and derivatives of the QS we implement this check as part of a line search procedure using the simple condition

$$s(\mathcal{P}_D[\theta + \alpha h]) < s(\theta) - \epsilon \tag{3.11}$$

of decreasing function values coupled with a backtracking algorithm, where $\epsilon > 0$ is used to ensure a minimum improvement in each iteration similar to the *sufficient decrease* condition in the Goldstein-Armijo rule. Further, a backtracking line search is applied on the projection arc $\mathcal{P}_D[\theta + \alpha h]$ and hence all iterates stay within the feasable set $\mathcal{D}$.

3.3. Estimating the error of the quasi-score approximation

As a measure of accuracy of the QS approximation we transform the prediction variances into an approximation error of the QS function. To this end, let

$$\hat{B}_\theta = \hat{Z}'(\theta)\hat{V}(\theta)^{-1} \ \in \mathbb{R}^{q\times p} \tag{3.12}$$

be a non-random weighting matrix given the data $Y = y$ w.r.t. to P_θ. Then, assuming $\hat{B}_\theta$ and the *statistical error* of the QS function to be known, the QS approximation error reads

$$\hat{C}(\theta) = \mathrm{Var}(\hat{B}_\theta(y - \hat{Z}(\theta))) = \hat{B}_\theta\hat{\Sigma}_K(\theta)\hat{B}_\theta^t \ \in \mathbb{R}^{q\times q}, \tag{3.13}$$

where the variance operation is w.r.t. to the distribution of the kriging predictor $\hat{Z}$ and $\hat{\Sigma}_K$ denotes the diagonal matrix of kriging variances. Since the inverse of the QI matrix serves as an estimated statistical lower bound (like the Cramér-Rao bound in ML theory) on the estimation error of $\hat{\theta}_0$, we can compare the QS approximation error with the overall achievable accuracy given by the QI matrix in (3.2). More specifically, if the largest *generalized eigenvalue* (see, e. g. **?**), say $\lambda^{max} = \lambda^{max}(\theta)$, of these two matrices drops below some limit $c \ll 1$ such that

$$\hat{C} \ll_L c \cdot \hat{I} \tag{3.14}$$

holds in the Loewner partial ordering of non-negative definite matrices, we say, that the QS approximation error is negligible compared to the predicted error of $\hat{\theta}$. This can be seen as an indication that no further benefit can be expected (from sampling new points) since we cannot get any better in estimating the unknown model parameter given the simulated statistics at the current design.

3.4. Termination conditions

We define two types of termination conditions of the estimation procedure. First, the *statistical stopping conditions*, control the sampling process and measure the level of accuracy of the QS approximation and, secondly, the *numerical conditions* which heuristically monitor the convergence of the sequence of approximated roots of the QS or stationary points of the criterion function. The algorithm stops sampling new evaluation points as soon as any of these conditions hold and immediately terminates the estimation.

The statistical conditions measure the improvement of the QS approximation achieved at iteration n. Since any criterion for stopping may strongly fluctuate from one iteration to another due to the simulation variability, we require some of them to hold for at least a number of consecutive iterations wich can be specified as an input argument to the algorithm. More specifically, we compare the approximation error of the QS to its overall achievable accuracy measured by the estimated QI matrix via the maximum generalized eigenvalue, say λ_n^{max}. If $\lambda_n^{max} < \lambda_{tol}$ or $\lambda_n^{rel} < \lambda_{rel}$ hold for at least N^{λ}, respectively, $N^{\lambda_{rel}}$ consecutive iterations we stop sampling. Both conditions relate to the fact of only approximately knowing the QS vector based on the assumed stochastic model within a black box treatment of the involved expectations and variances. We immediately stop the main iteration of the allgorithm if any of the above termination condition holds.

In order to monitor the numerical convergence of the iterates, let

$$Q_{max}(\theta) = \max_{1 \le i \le q} \{|(\hat{Q}(\hat{\theta}))^{(i)}|\}$$

denote the largest absolute deviation of the quasi-score vector from zero. Numerical convergence, besides others, is signaled by the conditions $Q_{max}(\theta) < \tau_0$ and $s(\theta) < s_{tol}$, where $s_{tol} > 0$ and $\tau_0 > 0$ are user-specified constants. For sufficiently small constants s_{tol}, τ_0 both conditions indicate an approximate root of the QS vector. In particular, the latter also allows for a statistical interpretation of the quasi-deviance as part of a Monte Carlo approach for testing the goodness-of-fit of the statistical model w.r.t. the estimated parameter $\hat{\theta}$ and provides us with another statistical stopping condition (see Section 4.7).

4. Extensions and modifications

This section presents some extensions to the originally proposed QL approach in **?**. Although it gives some insight in the algorithmic structure of the method the section can be skipped.

4.1. Select local evaluation points

Let

$$\mathcal{S}_n = \{\theta_1, \ldots, \theta_n\} \subset \mathcal{D}. \tag{4.1}$$

denote the sampling set of points of size n. The idea is to consider the value of the QD as a test statistic and hence using it as a decision rule to switch between the global and local phase of the algorithm. Suppose the algorithm has either found a solution θ^* to the QS equation in (3.6) or a (local) minimizer of the QD in (3.9) at some iteration $n \geq n_0$ based on the current sampling set $\mathcal{S}_n$. If the value of the QD function at θ^* is small compared to a user-defined upper bound, then θ^* is considered as an estimate of the model parameter and the algorithm stays in its local phase exploring the local vicinity for further refinement of θ^*. During both phases the algorithm samples new points of $\mathcal{D}$ which we call *candidates* for evaluation. In particular, during the local phase the algorithm randomly generates candidate points (without evaluating them) $\theta \in \Omega_n^l$, where Ω_n^l denotes the set of local candidates, according to a multivariate normal distribution with mean equal to the current minimizer and variance equal to the inverse information matrix in (3.2). Otherwise, the algorithm is in its global phase.

In order to measure the potential benefit of adding some point out of all candidates in Ω_n^l during the local phase of the algorithm we adapt the idea proposed in **?** to our setting. Let

$$\bar{s}_n(\theta) := \hat{Q}_n(\theta)^t \hat{C}_n(\theta)^{-1} \hat{Q}_n(\theta) \tag{4.2}$$

denote a *modified* version of the QD in (3.3), that is, the quasi-information matrix replaced by the error matrix of the QS vector (see Section 3.3) at iteration n. Then

$$\tilde{s}_n(\theta) = \begin{cases} \frac{\bar{s}_n(\theta) - \bar{s}_n^{\min}}{\bar{s}_n^{\max} - \bar{s}_n^{\min}} & \text{if } \bar{s}_n^{\min} \neq \bar{s}_n^{\max} \\ 1 & \text{otherwise.} \end{cases} \tag{4.3}$$

denotes its normalized value where the minimum and maximum of the modified QD is given as $\bar{s}_n^{\min} = \min\{\bar{s}_n(\theta) \,:\, \theta \in \Omega_n^l\}$ and, respectively, $\bar{s}_n^{\max} = \max\{\bar{s}_n(\theta) \,:\, \theta \in \Omega_n^l\}$. Also let

$$\Delta_n(\theta) = \min_{\tilde{\theta} \in \mathcal{S}_n} ||\theta - \tilde{\theta}||_2$$

denote the minimum distance of points $\theta \in \mathcal{D}$ to all previously sampled points belonging to $\mathcal{S}_n$. Then, the maximum and minimum values of these distances over all pairs of n points in $\mathcal{D}$ are given by

$$\begin{aligned} \Delta_n^{\min} &= \min\{\Delta_n(\theta) \,:\, \theta \in \mathcal{D}\}, \\ \Delta_n^{\max} &= \max\{\Delta_n(\theta) \,:\, \theta \in \mathcal{D}\}. \end{aligned}$$

Again, we normalize the distances $\Delta_n(\theta)$ to the interval $[0, 1]$ by

$$w_n(\theta) = \begin{cases} \frac{\Delta_n^{\max} - \Delta_n(\theta)}{\Delta_n^{\max} - \Delta_n^{\min}} & \text{if } \Delta_n^{\min} \neq \Delta_n^{\max} \\ 1 & \text{otherwise.} \end{cases}$$

Then, the next point which minimizes

$$v_n(\theta) = \gamma_j^l \tilde{s}_n(\theta) + (1 - \gamma_j^l) w_n(\theta) \tag{4.4}$$

is added to $\mathcal{S}_n$ for evaluation, where $1 \leq j \leq r$ and $\theta \in \Omega_n^l$. For given weights $0 \leq \gamma_1^l, \ldots, \gamma_r^l \leq 1$ we refer to $\Gamma^l = \langle \gamma_1^l, \ldots, \gamma_r^l \rangle$ as a *local weight pattern* of *cycle length* r. The weights define the degree of global/local selection of new evaluation points. More specifically, this local

selection criterion results in minimizing the modified QD if $\gamma_r^l = 1$ and otherwise, if $\gamma_1^l = 0$, maximizing the minimum distance to all other sampling points from $\mathcal{S}_n$. Cycling through Γ^l selects new points which have relatively low values of the modified QD and become more distant to previously sampled points for increasing weights. Based on the newly simulated statistics at the selected point the kriging models are updated and thus the QD approximation is further improved as represented in the last block at the bottom of Figure 1. The process is iterated until any stopping condition (see Section 3.4) is satisfied. The resulting solution based on the final approximation of the QS and QD is taken as an estimate of the unknown model parameter.

Besides a pre-specified weight pattern the algorithm also includes an automated adjustment of the weights according to the progress of the iterates and achieved accuracy of the QS approximation. By this, we intend to faciliate convergence of our method. We say that the current local phase at iteration n was *successful* in case the following holds

$$s_n < s_{n-1},\ \lambda_n^{max} < \lambda_{n-1}^{max}$$

and *failed* otherwise. We record the number successful and failed local phase iterations and as soon as the above conditions hold for a certain number of consecutive iterations the weights are increased by a user-defined scalar value. Otherwise the weights are decreased in the same way. We follow the reasoning that for a sequence of increasing weights the estimated accuracy of sequential QS approximations improves locally and thus can be trusted more and more so that new evaluation points are selected in the vicinity of the current best estimate. This forces the algorithm to select and evaluate new points more close to the current best estimate in order to reduce the (approximately pointwise) simulation variability of the statistics and hence to improve the accurary of the QS at this point. Otherwise, for descreasing weights the algorithm tends to select points in between previously sampled points due to the higher weighted minimum-distance term in (4.4). This implies a less beneficial impact on imroving the QS approximation by previously evaluated points and, therefore, the algorithm follows the "infill" strategy.

As another local selection criterion we propose a distance-weighted version of the *trace criterion*, see **?**. The next sample point is selected as

$$\theta_{n+1} = \arg\max_{\theta \in \Omega_n^g} w_n(\theta) \frac{1}{q} tr\{\hat{C}_n(\theta)\},$$

where $tr\{\cdot\}$ denotes the trace of a matrix. The criterion mostly selects points in relatively unexplored regions of the parameter space with a relatively high approximation error of the QS function. However, potential candidates more close to already sampled points get smaller weights which prevents the algorithm from clustering too much around previously sampled points. This strategy might be useful in case we rapidly want to improve the quality of the QS approximation within a few additional simulations of the statistical model followed by a root finding local search. Note that this selection criterion has a tendency to sample points at the "border" of the sample space first due to the extrapolation properties of the kriging estimators.

4.2. Select global evaluation points

The main goal of the global phase of the algorithm is to sample new evaluation points which can be far away from previously sampled points and widely spread over the entire search

space. While the global sampling can improve the QS approximation at any point finding possible solutions anywhere in the search space the local sampling procedure essentially is a local search algorithm since it only explores a local region in the vicinity of the estimated solution and improves its predicted location. Therefore the local phase cannot be used for global optimization alone. The main reason why it is necessary to use a global sampling procedure is that we need to investigate different local regions where low values of the QD could turn out to be approximate solutions and the other way round, approximate roots could vanish while new sample points are evaluated. The two sampling procedures differ in the generation of candidate points and also in the selection of weights to achieve a balance between global and local search which is a basis for efficient global optimization.

Let Ω_n^g be the set of uniformly distributed global candidate points over $\mathcal{D}$. New evaluation points are selected from Ω_n^g based on a weighted distance measure, where candidate points with low predicted QD value obtain higher weights. In this way, the QD serves as a monitor function to indicate promising and relatively unexplored regions. We use a strictly positive weight function, that is, $u_n(\theta) > 0$ for all $\theta \in \Omega_n^g$, which never fully ignores any region of the search space. The form of the weight function, which also can be found in **?** as part of a different selection criterion, is defined by

$$u_n(\theta) = \exp\Big\{ -\gamma\, \tilde{s}_n(\theta)\Big\}, \tag{4.5}$$

where $0 \leq \gamma < \infty$ denotes a weight parameter and $\tilde{s}_n$ denotes the normalized QD in (4.3). We select the candidate which maximizes the weighted minimum distance

$$\theta_{n+1} = \arg\max_{\theta\in\Omega_n^g} u_n(\theta)\Delta_n(\theta) \tag{4.6}$$

for the next evaluation. The weight parameter controls the balance between a local and more global selection of new points and the weight function is constructed such that, if $\gamma = 0$, then $u_n = 1$ holds for all $\theta \in \Omega_n^g$. In this case, the value of the QD is ignored and the next sample point maximizes the miniumum distance, i. e. in a space-filling manner, from all other samples of $\mathcal{S}_n$. For parameter values $\gamma \to \infty$, candidate points with relatively low values of the QD obtain much higher weights and hence we more and more trust the approximation of the QD. As in the local phase, we also intend to balance the sampling and, therefore, switch between a local and more global sampling by iterating the weight parameters $\gamma \in \Gamma^g$ in the same way as in (4.4). In contrast to the local selection rule there is no automated update of these weights implemented.

Note that, besides the above infill selection criteria, which keep a balance between global and local search steps, we implicitly account for the approximation error of statistics in both criterion functions. As a result, even if the chosen weights lead to a pure minimization of one of the criterion functions, new candidates make a compromise between being a local minimizer and improving the accuracy of the QS approximation. Thus, pure local candidates still improve the QS function w.r.t. its approximation error.

4.3. Variance matrix estimation

Besides the kriging approach to approximate the variance matrix proposed in **?** the package additionally offers a (weighted) average approximation. Let $\theta_1, \ldots, \theta_n$ belong to the current sample set $\mathcal{S}_n$ at iteration n of the algorithm. The idea is to estimate $V(\theta)$ for $\theta \in \mathcal{D}$ by

averaging over the simulated sample of covariance matrices $V(\theta_1), \ldots, V(\theta_1)$. To this end, we estimate $V(\theta)$ by the mean sample covariance matrix $E(V_i)$, see **?**, of sample matrices $V_i = V(\theta_i)$, $i = 1, \ldots, n$, with an assumed i.i.d. scaled Wishart distribution for $V_i = V(\theta_i)$. The package includes two basic types of sample average estimators. The first one is based on the Cholesky reparametrization, i. e. matrix decomposition, of sample covariance matrices, wich reads

$$V = LL^t \quad \text{with} \quad L = \text{chol}(V) \in \mathbb{R}^{p \times p}, \tag{4.7}$$

where L is a lower triangular matrix with real and positive diagonal elements. Then, an estimator of $E(V_i)$ is given by

$$\bar{V}_C = \bar{L}\bar{L}^t, \text{ where } \quad \bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i \tag{4.8}$$

with $L_i = L(\theta_i)$. Alternatively, an estimator based on the matrix logarithm (see, e. g. **?**), i. e. $\log V = U(\log \Lambda)U^t$, of sample covariance matrices $V = U\Lambda U^t$ using the common spectral decomposition with an orthogonal matrix U, can be obtained by

$$\bar{V}_L = \exp\left\{\frac{1}{n}\sum_{i=1}^{n} \log V_i\right\}. \tag{4.9}$$

? empirically shows that both estimators perform well in a series of test problems.

However, a locally weighted version of the estimated mean sample covariance matrix in (4.8) or (4.9) might be preferable in case of an already available approximate root $\hat{\theta}_0$ by $Q(\hat{\theta}_0) \approx 0$ in (2.1). Therefore, we apply a Nadaraya-Watson *kernel-weighted average*, see **?**, for both types of averaging matrices. In analogy to (4.8), this leads to the estimator

$$\tilde{V}_C = \tilde{L}\tilde{L}^t \quad \text{with} \quad \tilde{L} = \frac{\sum_{i=1}^{n} K(\theta_i, \hat{\theta}_0) L_i}{\sum_{i=1}^{n} K(\theta_i, \hat{\theta}_0)}, \tag{4.10}$$

where K is the multivariate Gaussian kernel

$$K(\theta, \hat{\theta}_0) = \exp\left\{ -(\hat{\theta}_0 - \theta)^t W(\hat{\theta}_0)^{-1}(\hat{\theta}_0 - \theta)\right\} \tag{4.11}$$

and $W = \hat{I}^{-1}$ is the weighting matrix. Likewise, the weighted version of (4.9) reads

$$\tilde{V}_L = \exp\{\bar{M}\} \quad \text{with} \quad \bar{M} = \frac{\sum_{i=1}^{n} K(\theta_i, \hat{\theta}_0)(\log V_i)}{\sum_{i=1}^{n} K(\theta_i, \hat{\theta}_0)}, \tag{4.12}$$

where $\exp\{\cdot\}$ denotes a matrix exponential. Moreover, the choice of kernel function is motivated by the idea that the asymptotic distribution of $(\hat{\theta} - \theta)$ is (under conditions ensuring asymptotic normality of the statistics, like ergodicity and sufficient regularity) normal with variance I^{-1}, see **?**, sect. 4.3. Note that the weighted variance matrices in (4.10) and (4.12) depend on an approximate root $\hat{\theta}_0$ in (4.11), which also must be given in order to compute the variance estimate.

Finally, to account for the simulation error of $\hat{Z}$ we add $\hat{\Sigma}$ in (3.13) to the variance matrix average approximation by

$$\hat{V}(\theta) = V^* + \hat{\Sigma}(\theta), \tag{4.13}$$

where V^* stands for any of the above types of matrix estimates and $\hat{\Sigma}$ denotes the diagonal matrix of prediction variances of $\hat{Z}$. Alternatively, a cross-validation approach (see Section 4.4) can be applied to estimate the prediction variances. The same strategy is used for the originally proposed kriging approach to approximate the variance matrix in **?**.

4.4. Alternative estimation of prediction errors

The use of kriging models allows the construction of relatively inexpensive approximations of the sample means of statistics compared to the effort required for evaluating the statistics based on the statistical model simulations only. In addition, we have some readily available indication of prediction accuracy at unsampled points measured by the kriging predictor variances. However, the main drawback of considering the kriging variance as a measure of prediction uncertainty stems from its inherent independence of the observed values as part of the prediction. The kriging variance, therfore, is strongly related to the spatial configuration (locations and interdistances) of sampling points and thus should not be viewed as an predictor's precision but rather as an "precision indicator" as argued in **?**, sect. 3.4.4. Moreover, the kriging variance is directly affected by the subjectively chosen covariance structure which is nearly impossible to be correctly specified in practice and, therefore, simply set beforehand assumed to be known without error. Practically, the kriging predictor is obtained by simply plugging-in the estimated parameters of the covariance model for the observed data which formally leads to a conditional predictor variance. Then, for instance, the kriging variance might not reflect the total amount of uncertainty in predicting values at unsampled points and is often underestimated (see **?**, for a comprehensive discussion). Consequently, any algorithm for parameter estimation which relies too much on prediction variances could be seriously misleaded.

For this reason, the package also includes an alternative strategy to predict the actual error. We use a cross-validation (CV) approach to estimate the degree of dependence between the spatial location of sampled points including the response of the statistical model, i. e. the value of a statistic, and its impact on the level of accuracy for predicting at new points. In order to assess the validity and uncertainty of kriging approximations we consider the *jackknife variance* (based on a specific CV approach tailored for simulation-based kriging) as proposed in **?** as well as the estimation of the *root mean squared error* (RMSE), see **?**, including an option to control the computational effort. Note that we consider the prediction variance based on kriging and CV as complementary to one another and point out that the computation of both types does not require additional simulations of the statistical model.

Let

$$\mathcal{Y}_n = \{\bar{y}_1, \ldots, \bar{y}_n\} \tag{4.14}$$

denote the set of simulated sample means of statistics at parameters $\theta \in \mathcal{S}_n$ for $i = 1, \ldots, n$. Given the sampling set $\mathcal{S}_n$ as in (4.1) we eliminate the ith element of $\mathcal{S}_n$ and obtain the CV sample $\mathcal{S}_n^{(-i)}$ of size $n-1$. For completeness we present the formulas for the *jackknife's pseudo-value*, *jackknife variance* and refer to **?** for a detailed description. The jackknife's pseudo-value for the kriging predictor $\hat{Y}$ at some parameter θ is treated as a (kriging) mean based on the original sample $\mathcal{S}_n$ defined by

$$\tilde{y}_i(\theta) = n\hat{Y}(\theta) - (n-1)\hat{Y}^{(-i)}(\theta) \in \mathbb{R}\,, \tag{4.15}$$

where $\hat{Y}^{(-i)}(\theta)$ denotes the response of the kriging model at θ with the simulated data $(\theta_i, \bar{y}_i)$

removed from $\mathcal{S}_n \times \mathcal{Y}_n$ for $i = 1, \ldots, n$. From these values the jackknife variance at the sample point θ is estimated by the usual sample variance

$$\tilde{\sigma}^2_{CV}(\theta) = \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} (\tilde{y}_i(\theta) - \overline{\tilde{y}}(\theta))^2 \right) \quad \text{with} \quad \overline{\tilde{y}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \tilde{y}_i(\theta). \tag{4.16}$$

As another measure to assess the kriging model fidelity, for example, to validate different realizations of initial sampling designs, we can use

$$RMSE(\theta) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\hat{Y}^{(-i)}(\theta) - \hat{Y}(\theta) \right)^2} \tag{4.17}$$

at the sample point θ based on the CV approach mentioned before. Note that the calculation of the jackknife variance in (4.16) and RMSE for various unsampled points in a sequential selection procedure requires the re-estimation of covariance model parameters for each sampling set $\mathcal{S}_n^{(-i)}$. In practice, this could be prohibitive as the number of overall sampling points grows during the estimation procedure. The high computational effort is mainly due to the number of REML estimations which must be carried out for n covariance models each based on $n-1$ sampling points. If n becomes large, then the pseudo-values in (4.15) might not much differ anymore which would lead to variances mostly due to simulation error. Therefore, we also can apply a *block jackknife* scheme (see, e. g. **?**, and references therein) in addition to the *delete-1 jackknife* method for large n and proceed as follows. We limit the number n_c of covariance models still to fit, i. e. the number of subsets of $\mathcal{S}_n$, by $n_c \leq n$ with maximum k sampling points deleted from $\mathcal{S}_n$ for each covariance model fit such that $n = n_c k$. Then, we define the pseudo-values in analogy to (4.15) as

$$\hat{y}_i(\theta) = n_c \hat{Y}(\theta) - (n_c - 1) \hat{Y}^{(-i)}(\theta), \tag{4.18}$$

where the index i with $1 \leq i \leq n_c$ now refers to the block i, possibly being a one-element (if $k = 1$), which consists of k sample points to be removed from $\mathcal{S}_n$ for the purpose of covariance fitting and construction of the corresponding kriging predictors $\hat{Y}^{(-i)}$. Then, we proceed as in (4.16) or (4.17) with n replaced by n_c. Note that this case also includes the delete-1 jackknife method and thus we will always refer to n_c as the maximum number of blocks. Moreover, in **?** the authors report that choosing $k = 0.1n$ or $k = \sqrt{n}$ for the RMSE in (4.17) provides good error estimates when kriging models are used for prediction and design analyses. Therefore, we choose k proportional to the overall number of sample points n currently reached and dynamically increase k in a step by step manner if n grows beyond a pre-defined upper limit during the estimation procedure. Nevertheless, using prediction variances based on CV instead of kriging is obviously computationally more demanding since, despite of the strategy explained above, additional covariance models have to be estimated and iteratively updated during the overall estimation procedure. This is the price the user has to pay for a possibly more realistic treatment of prediction uncertainty which is likely to result in a more robust final model parameter estimate. Unless the structure of the general estimation problem is relatively simple we recommend to use the CV based approach.

4.5. Initial design validation

In order to contruct a reasonable global approximation model of the QD we show how to assess the predictive quality and goodness-of-fit of a pilot design based on each kriging model separately. The idea is to use the same type of prediction variance, e. g. by kriging or CV, for the analysis of the initial design and later on during the sequential candidate selection strategy. This ensures a consistent treatment of selecting a new candidate point without spending too much effort on the calculation of prediction variances or performing additional simulations.

The most commonly applied concepts are based on kriging prediction variances and resampling methods such as bootstrapping and cross-validation. The reader is refered to **?** for an overview of various strategies empirically investigated in the context of simulation-based optimization. Although, in principle, automated procedures are available to choose and augment a design optimally for a global metamodelling (see e. g. **?**, sect. 6) we rather focus on which type of prediction variance best reflects the true variability of predictions and hence the statistical model. The reasoning is, that this choice has a certain impact on the efficiency of the sampling process towards a reasonable estimate of the model parameter, because the prediction variance is directly involved in the selection of new evaluation points based on one of the available criterion functions. Besides this, a two-stage design optimization criterion, such as *integrated mean squared error* (see, e. g. **?**), could be applied.

Although, our estimation method does not require a specific initial design (as long as it inherits a space-filling property) a carefully chosen pilot design usually results in faster convergence of the algorithm and a less number of additional sample points. Instead of using CV for estimating the prediction variances at candidate points (see Section 4.4) here we use the CV approach to empirically validate the adequacy of the kriging model, for example, assumptions either about the employed class of covariance models and the involved fitted parameters, the initial sampling size used to build the kriging model or about the variability of simulated statistics over the entire parameter space. To this end, we modify the CV approach in **?**, sect. 11 as follows.

Let $\hat{Y}^{(-i)}(\theta_i)$ be the value of the kriging predictor at $\theta_i \in \mathcal{S}_0$ with the simulated data $(\theta_i, \bar{y}_i)$ removed from $\mathcal{S}_0 \times \mathcal{Y}_0$. If the average CV error (ACVE) defined by

$$\text{ACVE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}(\theta_i) - \hat{Y}^{(-i)}(\theta_i)\right) \approxeq 0\,, \tag{4.19}$$

where $\hat{Y}(\theta_i)$ denotes the prediction including the full sample set $\mathcal{S}_0$, $i = 1, \ldots, n$, then we say that there is no apparent bias in predicting at left-out sample points. Otherwise a value which significantly distinguishes from zero could represent (systematic) over- or underestimation of the kriging prediction variances by the corresponding predictor. Due to the simulation error of evaluating the statistics Y we estimate the observed value $\bar{y}_i$ by the noise-free response, see **?**, sect. 3.7.1, of the kriging predictor in (4.19) instead of the simulated statistics itself as proposed in **?**.

Also, the magnitude of the *mean squared cross-validation errors* (MSE) of predicting the statistics gives an impression on the sensitivity of the corresponding statistic Y leaving out a sample point i of the design for estimating the corresponding sample mean value

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}(\theta_i) - \hat{Y}^{(-i)}(\theta_i)\right)^2. \tag{4.20}$$

Further, to compare the magnitudes of the actual CV error with the one induced by the corresponding kriging model we define the squared *standardized cross-validation error* (SCVE), which reads

$$\mathrm{SCVE}(\theta_i) = \frac{\left(\hat{Y}(\theta_i) - \hat{Y}^{(-i)}(\theta_i)\right)^2}{(\hat{\sigma}^2)^{(-i)}(\theta_i)}\,, \tag{4.21}$$

where $(\hat{\sigma}^2)^{(-i)}$ denotes the kriging variance with the data pair $(\theta_i, \bar{y}_i)$ omitted from the sample set $\mathcal{S}_0 \times \mathcal{Y}_0$. Then, the average of the SCVE

$$\mathrm{ASCVE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{SCVE}(\theta_i) \tag{4.22}$$

can be used to empirically compare kriging models based on different sampling designs and/or covariance models w.r.t their adequacy and validity of covariance parameter estimates. If we have ASCVE $\cong 1$, then the CV errors approximately equal on average the ones predicted by the kriging variance. Given a particular pilot design this suggests a good match of both types of errors and hence the kriging model appropriately models the simulation output. Otherwise, if ASCVE $\gg 1$, i. e. being significantly greater than one, we would be in favour for using CV errors for a more realistic treatment of the prediction uncertainty during the sequential sampling procedure and, in case of ASCVE $\ll 1$, prefer kriging prediction variances. If the initial design does not meet the users requirements up to some level of adequacy according to (4.22) or the approximate unbiasedness property (4.19), then we can easily augment the current design by the function `augmentLHS()` from the package **lhs** (**?**) or by `multiDimLHS()` as part of our package. In this case, additional simulations at these newly generated design points hopefully provide more information and might improve the quality of the initial kriging model. These steps could be repeated until the user is satisfied with the results. Note that this procedure is used to assess the predictive performance of each kriging model separately since these are based on individual estimated covariance functions.

Apart from this, the package also includes functions to explicitly evaluate the kriging variances, e. g. `varKM()`, and predicted CV errors, like `crossValTx()`, of each kriging model. These functions are intended to provide a basis for a comprehensive analysis of each kriging model in order to apply other (user-defined) approaches to improve the predictive quality depending on the (simulated) data and initial design.

4.6. Numerical consistency of solutions

If the algorithm terminates successfully with an approximate root $\hat{\theta}$, we employ the method proposed in **?**, sect. 13.3.3 for examining its consistency as a solution to the approximating QS. The same principles apply having found multiple roots as candidates of the true model parameter.

Let $\hat{Q}'$ denote the matrix derivative of the QS vector, i. e. the *observed quasi-information* (see **?**, sect. 13.2). We assume that $\hat{Q}'(\theta) \sim E_{\theta_0}(\hat{Q}'(\theta))$ in probability for $\theta \in \Theta$. Then, if $\hat{\theta}$ is a consistent estimator of θ_0,

$$E_{\hat{\theta}}(\hat{Q}'(\hat{\theta}))^{-1}\hat{Q}'(\hat{\theta}) \sim Id \tag{4.23}$$

in probability, where Id is the $q \times q$ identity matrix. Since $\hat{Q}$ is a standardized QS for which the *score property*

$$E_{\hat{\theta}}(-\hat{Q}') = E_{\hat{\theta}}(\hat{Q}\hat{Q}^t), \tag{4.24}$$

see **?**, sect. 13.3.1, holds by (2.3) and (2.4), we identify $\hat{\theta}$ as the correct root if the ratio $\hat{I}(\hat{\theta})^{-1}\hat{Q}'(\hat{\theta})$ approximately equals the identity matrix. For this, we first check the positive definiteness of $-\hat{Q}'(\hat{\theta})$ for $\hat{\theta}$ being a consistent estimator of the true parameter value θ_0, which, in case of ML estimation, corresponds to a maximum of the likelihood function. Let

$$M = \hat{I}(\hat{\theta})^{-1}\hat{Q}'(\hat{\theta}),$$

then we can inspect (4.23) in terms of its numerical properties by:

$$\begin{aligned} det_M &= |1 - det(M)|, \\ max_M &= \max_{1\leq i\leq q}\{|1 - M_{ii}|\}, \\ tr_M &= |1 - tr\{M\}/q|. \end{aligned} \tag{4.25}$$

For $\hat{\theta}$ to be a consistent root of $\hat{Q}(\hat{\theta})$ (at least numerically) we require to hold $det_M < \tau_1$, $max_M < \tau_2$ or $tr_M < \tau_3$, where τ_k are chosen constants sufficiently small and $k = 1, 2, 3$. In case of multiple roots it is reasonable to choose the one for which (4.25) yields smallest values. Finally, a goodness-of-fit test (see Section 4.7) can be applied in order to select the best root in a probabilistic sense.

4.7. Monte Carlo hypothesis testing

We implement a Monte Carlo (MC) hypothesis test (see, e. g. **?**, sect. 7.1) to assess the goodness-of-fit of the user-defined statistical model w.r.t. to the estimated model parameter. Following our general assumption that we only have a single or a few raw data (i. e. real observations $Y = y$) of identical situations available, while simulation replications obtained from the statistical model are relatively inexpensive to generate, we can estimate the variability of the model outcome (including any function of the model parameter) much better than we could in case of only measuring the real outcome of the observed statistics. Therefore, independent realizations of the simulated model, where the number of simulations is only limitted by the computational budget, allow us to assess the model variability and hence the contruction of a MC hypothesis test. The (conceptual) null hypothesisis is whether the data, e. g. observed values of the statistics, can be explained by the outcome variability of the model simulations.

The basic idea is to derive the MC version of the hypothesis test as proposed in **?** based on the simulated QS estimating function. Suppose that we have the following QL estimator $\hat{\theta}$, that is, a solution to the approximating quasi-score, $\hat{Q}(\hat{\theta})$, in our setting. We consider testing the following null hypothesis

$$H_0: \quad \hat{\theta} = \theta_0$$

against the alternative

$$H_1: \quad \hat{\theta} \neq \theta_0,$$

where θ_0 is the true model parameter. We either use the QD or some version of the MD as a test statistic which we denote by $S = S(\theta, X)$. Since S is random as a function of the (observed) model outcome $x = X$ and the parameter θ, in our setting, it can only be

evaluated by simulating the statistical model. As an approximation to the so-called *efficient score statistic* (**?**) it does not involve the evaluation of an objective function, for example, a likelihood. Under appropriate conditions the efficient score statistic approximately follows a chi-squared distribution, χ^2_q, with q degrees of freedom (see, e. g. **?**, sect. 9.2) and hence can be used to set up a hypothesis test about the true value θ_0. This test only depends on the restricted class of parameters under the null hypothesis and, therefore, only requires the test statistic to be evaluated at $\hat{\theta}$. However, the true probability distribution of S is unknown and follows the "randomness" generated by the statistical model given the data. Therefore, we construct a MC test analogue of the efficient score test as follows.

We proceed in three steps (see, e. g. **?**, pp. 53-56) in order to estimate the corresponding P-value:

1. simulate a large sample $X^*_1, \ldots, X^*_m \sim \mathrm{P}_{\hat{\theta}}$ of m independent observations of the statistical model under the null hypothesis,
2. re-estimate the model parameters $\hat{\theta}^*_1, \ldots, \hat{\theta}^*_m$ given X^*_j, $j = 1, \ldots, m$,
3. compute the test statistics $s_j = S_j$, where $S_j = S(\hat{\theta}^*_j, X^*_j)$, for each jth model outcome

Suppose that $s_{obs} = s(\hat{\theta}, x)$ is the observed value of the test statistic given the data, i. e. $y = T(x)$, and the estimated model parameter $\hat{\theta}$. Then, we obtain a MC estimate of the (upper tail) P-value of the test statistic by

$$\hat{P} = \hat{\mathrm{P}}(S \geq s_{obs}) = \frac{\sum_{j=1}^{m} \mathbf{1}_{[s_{obs},\infty)}\{s_j\} + 1}{m + 1}$$

as the proportion of values s_j, $j = 1, \ldots, m$, based on the simulated data X^*_j under the null hypothesis, exceeding the observed value of the test statistic. If the observed statistics can be appropriately fitted by the statistical model, the value s_{obs} will not be significantly different from the simulated values of the test statistic. Further, in our implementation, we also allow the test statistic to be different from the criterion function which was first used to estimate the model parameter.

Note that, to be consistent, we re-estimate $\hat{\theta}^*$ (given the simulated observed statistics) by the same procedure and choice of criterion function as before when estimating the model parameter $\hat{\theta}$. The difference is that we merely use model realizations already generated for the estimation of $\hat{\theta}$ and by this only require additional simulations at $\hat{\theta}$ in order to compute the test statistic for each replicated data, i. e. model outcome. Obviously, using additional model simulations during the re-estimation would enforce the algorithm to produce more "precise" parameter re-estimates. However, such procedure would render our test infeasible since then the computational effort would be comparable to a full-fledged simulation study. The point is, if an estimate of the model parameter was found depending on the budget for simulating the statistical model, we assume that the search space is sufficiently explored by additional sample points up to a certain user-specified accuracy. Hence, in practice, the re-estimated paramters should then change not too much compared to using additional sample points and simulations. Finally, since we implicitely estimate the sampling distribution of $\hat{\theta}$ during the test, we can also calculate the standard error of the estimated parameter and derive further error measures based on the inverse QI matrix, which we call *predicted standard errors* (see the R examples below).

Optionally, this test procedure is also sequentially employed when promising local minimizers of the criterion function are visited during the local phase of the algorithm. Suppose that $\hat{\theta}$ is a local minimizer of the criterion function, then we use this test in order to assess the plausibility of $\hat{\theta}$ being a potential root of the corresponding quasi-score vector given the data $X = x$. New observations $(x_1^*, \ldots, x_m^*)$ are simulated w.r.t. to $\hat{\theta}$ and the algorithm re-estimates the approximate roots $\hat{\theta}_j$ for each observation x_j^*. If

$$s(\hat{\theta}; x) \leq \hat{F}_m^{-1}(1-\alpha) \tag{4.26}$$

holds, where $\hat{F}_m$ denotes the empirical c.d.f. of $s(\hat{\theta}_1; x_1^*), \ldots, s(\hat{\theta}_m; x_m^*)$ related to the test statistic S, then $\hat{\theta}$ is accepted as an approximate root at significance level α. In this case the algorithm stays in its local phase and continues sampling around the current root according to its asymptotic variance (measured by the inverse of the predicted quasi-information) and uses the additional simulation results to improve the current kriging approximations. Otherwise, the last evaluation point is used as a starting point for next local searches which mimics a random multistart type minimization of the criterion function over the next iterations. This approach also has the potential to escape regions where the criterion function value is quite low and, however, is not considered trustworthy with regard to the empirical quantile (4.26) of the test statistic.

We can also derive a statistical stopping condition (see Section 3.4 based on the above procedure. Since the variance of the QL estimator is under certain conditions asymptotically equivalent to the inverse of the quasi-information in (2.4) we define the relative efficiency (EF) in spirit of the coefficient of determination from linear regression by

$$\mathrm{EF} = \left| 1 - \frac{\left[\hat{I}(\hat{\theta})^{-1}\right]_{ii}}{\left[\frac{1}{m}\sum_{j=1}^{m}(\hat{\theta}-\hat{\theta}_j)(\hat{\theta}-\hat{\theta}_j)^t\right]_{ii}} \right| \tag{4.27}$$

for the comparison of the empirical estimation error of $\hat{\theta} \in \mathbb{R}^q$, $i = 1, \ldots, q$, with its predicted error (measured by the inverse of the quasi-information matrix) and stop as soon as both errors deviate less than a user-defined fraction from each other for a specified number of consecutive iterations. Further stopping conditions can be found in the R manual of the function `qle`.

5. Estimation with qle in R

The package includes functions for simulating a user-specified statistical model, estimating the unknown model parameter and performing a goodness-of-fit hypothesis test. Several options for parameter estimation are available which allow to choose different criterion functions as well as types of variance matrix approximations and prediction variances of the kriging estimator. Users can choose among different (initial) sampling designs, best candidate selection strategies and local or global (derivative free or gradient-based) minimization methods. Among these, the quasi-scoring algorithm estimates the model parameter as an approximate root of the QS vector or by a stationary point of one of the criterion functions proposed including simple bound constraints. Finally, the estimated model parameter can be tested by a Monte Carlo approach to hypothesis testing.

Apart from these methods related to parameter estimation, the package also offers access to low level functions, such as estimating the parameters of a particular covariance model (by the restricted maximum likelihood method) for predicting the sample means of the statistics or approximation of the variance of the statistics by kriging. We allow for different covariance models and specify how to incorporate the simulation variances of the statistics as so-called local or global nugget variances for each corresponding covariance model separately. We provide functions to inspect their impact on the predictive quality mainly by comparing individual prediction variances based on the CV approach and kriging models. This can be seen as a preliminary step to construct a reasonable pilot design for parameter estimation which then is sequentially improved.

5.1. Main functions

We start with a short overview of the main functions and exemplify the practical workflow. Note that all examples presented in this Section are also available as separate R source files.

First, the user must define a simulation function (simulating a given statistical model) which expects a numeric vector of statistical model parameters as its first argument and returns a numeric vector of the user-defined statistics in order to characterise the model outcome. Further arguments can be passed by '...' for all functions of the package where the simulation function is required as an input argument. An explicit naming of the parameter values or statistics is not required. Without supplied names we use the naming convention `T1`, `T2`, ... for the first, second and so on, statistic.

The main function for estimation is `qle()` which minimizes one of the criterion functions (e. g. quasi-deviance or different versions of the Mahalanobis distance), samples new evaluation points and returns an object of class `qle`. Other functions, for example, `qscoring()` or `searchMinimizer()`, only search for a root of the QS vector or, respectively, a minimizer of the chosen criterion function without using additional simulations of the statistical model.

Typically, the following functions from the package have to be called in the order of appearance in order to initialize the estimation method:

1. `multiDimLHS()`, generate initial design of space-filling points,
2. `simQLdata()`, simulate the statistical model at these design points,
3. `setQLdata()`, collect simulation results for QL estimation,
4. `fitSIRFk()`, fit a covariance model ('`sirfk`' by default); alternatively, each statistic could also be modelled by different covariance functions which then would require a manual setup (see Section 5),
5. `QLmodel()`, construct the QL approximation model of class `QLmodel`, that is, store the covariance models of statistics, the observed statistics, the initial design points, simulated values of the statistics as well as several options for local optimizations and parameter estimation.

The last three functions are wrapped up in a single function call by `getQLmodel()` for convenience, which also returns an object of class `QLmodel`. For reasons of simplicity, the initial

design can also be generated by `simQLdata()` and stored in the returned object. In addition, there are a couple of built-in convenience functions which help to analyse the predictive quality of each covariance model, for example, using kriging predictions of the sample means of the statistics with the help of `predictKM()`, evaluating the corresponding kriging or CV based prediction variances by `varKM()` or, respectively, `crossValTx()`.

5.2. Options for parallel processing

The package provides two built-in types of parallel processing including the stochastic model simulations based on the **parallel** package. Both types can be combined in a HPC environment using a number of compute nodes or multiple CPUs of a single host. First, computations can be run in parallel by spawning the computations to multiple CPU cores (see `mclapply`[3]) or distributing the computations (including model simulations) to different compute nodes of a HPC using a number of local CPUs at each node as a kind of nested parallel processing. The user can pass any valid cluster object of class `"MPIcluster","SOCKcluster","cluster"` which supports calling `parallel::parLapply()` to the **qle** and others of the package. Note that by default computations are processed sequentially using a single CPU/core. Please see the manual for details of all parallel options.

5.3. M/M/1 queue

As an introductory example we consider the following single-server queueing system denoted as M/M/1 (see, e. g. **?**). Let N represent the random variable "number of customers" in the system at steady state. Then N is geometrically distributed with success parameter $1-\rho$ and

$$\mathsf{E}[N] = \frac{\rho}{1-\rho}$$

is the expectation of N as a function of $\rho < 1$. Hence the parameter of interest for estimation is $\theta = \rho$, which can be interpreted as the fraction of time the server is working, and the variance of N is given by

$$\mathsf{Var}_\theta(N) = \frac{\rho}{(1-\rho)^2}.$$

Let $y = (y_1, \ldots, y_n)^t$ be the observed number of customers at n different time points where we use the sample mean $\bar{y} = \sum_{i=1}^{n} y_i/n$ as the summary statistic for estimation by our method. Thus from each replication of the statistical model we record the average number of customers in the system which, following the reasoning in Section 3, is approximated by a kriging surface, see Figure 2. We start by setting the options for parallel processing,

```
R> options(mc.cores=8L)
R> options(qle.multicore="mclapply")
R> RNGkind("L'Ecuyer-CMRG")
R> set.seed(1326)
```

and define the statistical model which practically leads to the simulation function `simfn` shown below. For estimation of the parameter ρ by our approach we fix the number of time points of observations to $n = 100$.

[3]Note that `mclapply` is not available on Windows-based platforms, however, then run sequentially by `lapply`.

```
R> cond <- list("n"=100)
R> simfn <- function(tet,cond){
+         mean(rgeom(cond$n,prob=1-tet[1]))
+ }
```

The function returns the average the number of customers in the system at steady state. Next, we set the lower and upper bounds of the parameter search space,

```
R> lb <- c("rho"=0.05)
R> ub <- c("rho"=0.95)
```

and sample the initial design points for constructing the approximation of the criterion function (including the summary statistic). At each design point `nsim` simulations are used to estimate the sample mean of the statistic.

```
R> nsim <- 10
R> X <- multiDimLHS(N=9,lb=lb,ub=ub,
+       method="maximinLHS",type="matrix")
R> sim <- simQLdata(sim=simfn,cond=cond,nsim=nsim,X=X)
```

We set the "real" observation of the statistic $Y = y$ to `obs=1` corresponding to the parameter $\rho = 0.5$. In this example the variance of Y is approximated by the average of the matrix logarithm of the sample variance matrices at each design point. We obtain the QL approximation model by

```
R> qsd <- getQLmodel(sim, lb, ub, obs=c("N"=1),
+                  var.type="wlogMean",verbose=TRUE)
```

and, as a first crude estimate of the model parameter, we apply the quasi-scoring iteration without using further simulations.

```
R> S0 <- qscoring(qsd,x0=c("rho"=0.8))
R> print(S0)

R> print(S0)

Local method:

 'qscoring'

Start:

    rho
0.80000

Solution:

    rho
```

```
0.51503

Quasi-deviance:

 4.13e-14

Iterations.... 4
Status........ 1 ( QFS_SCORETOL_REACHED )

 Optimization stopped because score_tol was reached.

Quasi-score:

[1]    -0.0000071702574
```

The function returns an estimated root already quite close to the "true" one. However, we can improve the current estimate sampling `maxeval` additional points each using at most `nsim` simulations. We choose the `score` criterion, see (4.4), for selecting the next evaluation points equally weighting the interdistances to previous sample points (as filling in the gaps) and the criterion function value by `weights=0.5`. As a local search, respectively, root finding method, we apply the quasi-scoring iteration and, in case of non-convergence, switch to the method `bobyqa` or even `direct` as a global search method directly appplied to the criterion function. Note that we also *test* a found minimizer whether it could be an approximate root of the quasi-score vector.

```
R> OPT <- qle(qsd,simfn,cond=cond,
+   global.opts = list("maxeval"=5, "NmaxLam"=5),
+   local.opts = list("nextSample"="score","weights"=0.5,
+                     "ftol_abs"=1e-4, "lam_max"=1e-5),
+   method = c("qscoring","bobyqa","direct"), iseed=1326)
```

Only a few iterations are needed to improve the last estimate.

```
R> OPT

Quasi-deviance:

 1.272e-08

Estimate:

    rho
0.50565

Convergence: maximum evaluations reached.

    maxeval
```

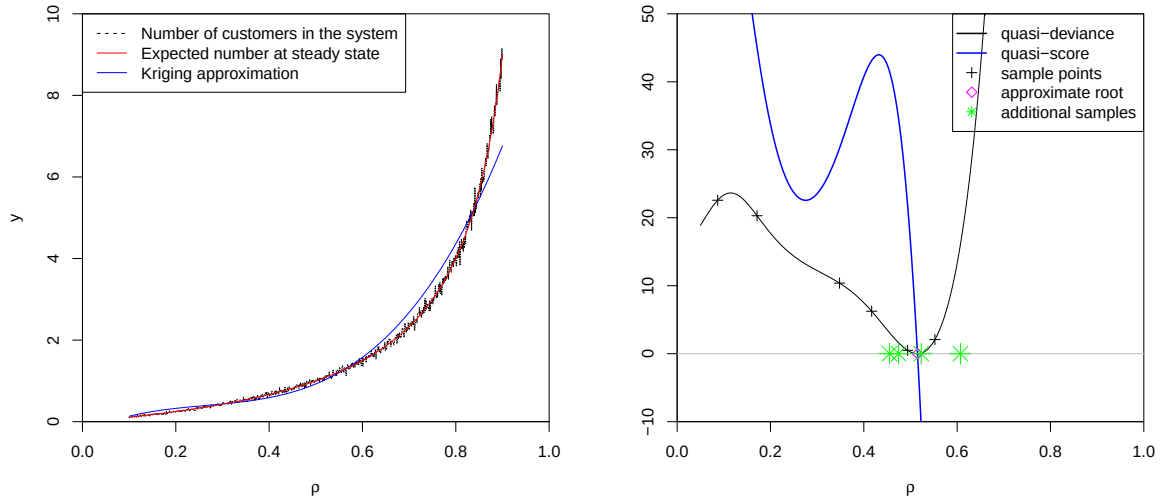

Figure 2: M/M/1 queue: number of customers (left) and quasi-deviance, respectively, quasi-score approximation (right). All approximations shown are based on the same set of evaluation points.

```
5.00000e+00

Evaluations:  5  (simulations:  50 )

Variance approximation type:  wlogMean
```

The results of the consistency criteria (see Section 4.6) are given below.

```
R> checkMultRoot(OPT,verbose = TRUE)

       rho minor     value |score_max|     |det|     |max|   |trace|
par 0.5057     0 1.272e-08    0.003397 0.0003291 0.0003291 0.0003291
```

Both the predicted, that is, the expected quasi-information matrix and its (numerically evaluated) observed analogue show a good aggreement. The results suggest a consistent root of the quasi-score and thus a plausible estimate of the unknown parameter.

Further, we compute the prediction of the summary statistic given the final design `X` and sample average values `Tstat` (at these points)

```
R> X <- as.matrix(OPT$qsd$qldata[,1])
R> Tstat <- OPT$qsd$qldata[grep("mean.",names(qsd$qldata))]
R> predictKM(OPT$qsd$covT,c("rho"=0.5),X,Tstat)

     mean.T1
[1,]  0.9776
```

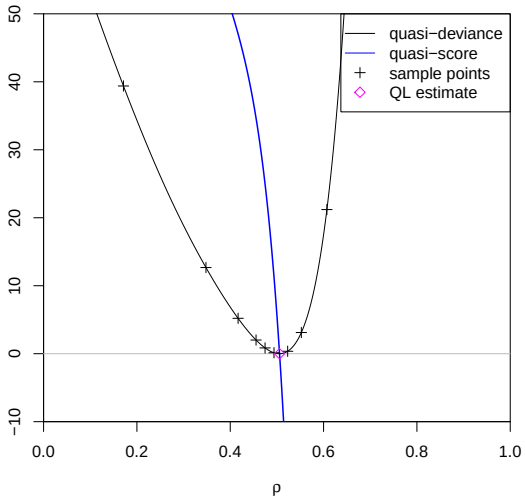

Figure 3: Final quasi-deviance and quasi-score function approximation after adding five new evaluation points.

Since the joint density function of the (simulated) observations y is known we can derive the *maximum likelihood estimator* (mle), which reads

$$\hat{\rho}_{mle} = 1 - \frac{1}{1+\bar{y}} \tag{5.1}$$

as the solution of Fisher's *score function*,

$$u(\rho) = n\left(\frac{1}{1-\rho} - \frac{\bar{y}}{\rho}\right) = 0\,,$$

and belongs to the exponential family of distributions in linear form. In this particular case the score function from MLE is identical to the quasi-score function,

$$Q(\rho,\bar{y}) = \frac{1}{(1-\rho)}\left(\frac{\rho}{n(1-\rho)^2}\right)^{-1}\left(\bar{y} - \frac{\rho}{1-\rho}\right) = u(\rho)\,. \tag{5.2}$$

Further, since the variance of $\hat{\rho}_{mle}$ is given by

$$\mathsf{Var}(\hat{\rho}_{mle}) = \frac{\hat{\rho}(1-\hat{\rho})^2}{n},$$

the same holds for the quasi-information matrix and Fisher's *information* matrix, which reads

$$I(\rho) = \frac{1}{(1-\rho)^4}\left(\frac{\rho}{n(1-\rho)^2}\right)^{-1} = \frac{n}{\rho(1-\rho)^2} = \mathsf{Var}(\rho)^{-1} = I_{mle}(\rho).$$

Therefore, compared to the maximum liklihood method, the resulting estimation error by quasi-likelihood estimation is due to the inherent simulation variance and approximation error of $\bar{y}$ induced by kriging the statistics. This effect can be best exemplified by a short simulation study as shown next.

We generate some observations at the "true" parameter, $\rho = 0.5$.

```
R> tet0 <- c("rho"=0.5)
R> obs0 <- simQLdata(sim=simfn,cond=cond,nsim=100,X=tet0)
```

and compute the empirical mean squared error (MSE) of the maximum likelihood estimates and the average variance over all estimated variances denoted by $\overline{Var}$.

```
R> mle <- do.call(rbind,
+                  lapply(obs0[[1]],function(y,n){
+                 tet <- 1-1/(1+y[[1]])
+                 c("mle.rho"=tet,"mle.var"=(tet*(1-tet)^2)/n)
+              }, n=cond$n))
R> x <- mle[,1]-tet0
R> mle.var <- c(sum(x^2)/length(x),mean(mle[,2]))
```

Given the generated set of observations `obs0` we now estimate the model parameter for each observation by quasi-likelihood

```
R> OPTS <- parLapply(cl,obs0[[1]],
+            function(obs,...) {
+             qle(...,obs=obs)
+            },
+          qsd=qsd,
+          sim=simfn,
+          cond=cond,
+          global.opts=list("maxeval"=10,"NmaxLam"=10),
+          local.opts=list("nextSample"="score","weights"=0.5,
+                          "ftol_abs"=1e-4,"lam_max"=1e-5,
+                          "useWeights"=TRUE),
+          method=c("qscoring","bobyqa","direct"))

R> # get results
R> QLE <- do.call(rbind,
+         lapply(OPTS,
+           function(x) {
+            c("qle"=x$par,"qle.var"=1/as.numeric(x$final$I))
+                  }))
R> y <- QLE[,1]-tet0
R> # MSE and average estimated variance of the parameters
R> qle.var <- c(sum(y^2)/length(y),mean(QLE[,2]))
```

Table 1 summarises the empirical and predicted estimation variances resulting from the above simulation study. For both methods the predicted average variances as shown by $\overline{Var}$ and the empirical MSEs compare well.

Finally, we apply the MC goodness-of-fit test (see Section 4.7) of the estimated parameter based on the observations `obs0` taken from the simulation study.

```
R> Stest0 <- qleTest(OPT,sim=simfn,cond=cond,obs=obs0,cl=cl)
```

Method	MSE($\hat{\rho}$)	$\overline{\mathsf{Var}}(\hat{\rho})$
qle	0.00135	0.00112
mle	0.00133	0.00124
Score test	0.00135	0.00121

Table 1: Empirical mean squared error (MSE) and average variance $\overline{Var}$ of the re-estimated parameters for 100 observations using the quasi-likelihood approach and maximum likelihood method.

```
R> print(Stest0)

Call:

qleTest(OPT, sim = simfn, cond = cond, obs = obs0, cl = cl)

Coefficients:

     Estimate  Std. Error  RMSE      Bias       Mean
rho  0.50565   0.036677    0.036701  -0.001756  0.5039


Bootstrap Score-test:

   s value      Pr(>s)
1.2724e-08     0.13187


Average quasi-score:

[1]  -0.001153


Predicted Std. Errors:

     Average   Estimate  EF
rho  0.034788  0.033206  0.09522
```

The result of the Score test using the same observations of the simulation study shows a good match of the empirical and predicted error measures. Note that the errors resulting from the Score test are not directly compareable with the ones shown in Table 1 since the test procedure does not use new evaluation points (and thus additinoal simulations of the model) during the re-estimation of the parameter given the simulated observations. Instead, it is solely based on the fixed design of evaluation points stored in the estimation results `OPT`.

5.4. Estimating the parameters of a normal distribution

This example shows how to estimate the mean and standard deviation (μ, σ) of a gaussian

random variable. Of course, we do this rather for pedagogical benefits since the method of choice is ML. However, its simplicity allows us to demonstrate the basic steps which are required to initialize the method and also apply for other more complex parameter estimation problems in general. Besides this, it helps to understand why the method might fail in situations where the chosen summary statistics are less informative.

We define a simulation function which simply draws 10 random numbers at parameter $\theta = (\mu, \sigma)$ using the median and the *mean absolute deviation* (from the median) as our informative statistics to characterise the model outcome.

```
R> # use the local cluster
R> cl <- makeCluster(8L)
R> clusterSetRNGStream(cl,1234)
R> ## Multicore parallel processing:
R> # options(qle.multicore="mclapply")
R> # options(mc.cores=2L)
R> simfunc <- function(pars) {
+         x <- rnorm(10,mean=pars["mu"],sd=pars["sigma"])
+         c("T1"=median(x),"T2"=mad(x))
+ }
```

The simple box constraints below, `lb` and `ub`, define the parameter search space as lower and, respectively, upper bounds of the unknown parameter of equal length.

```
R> lb <- c("mu"=0.5,"sigma"=0.1)
R> ub <- c("mu"=8.0,"sigma"=5.0)
```

We choose a *maximin design* as a pilot design in order to construct the initial approximation of the QS vector using the quasi-deviance as a criterion function. We initially sample `N` design points and simulate `nsim` times at each point.

```
R> sim <- simQLdata(sim=simfunc,
+    nsim=10,N=8,lb=lb,ub=ub,method="maximinLHS")
R> # reset number of simulations (10 x 10)
R> attr(sim,"nsim") <- 100
```

Although, in this example, we could simulate an observed value `obs` of the statistics we use the population mean and standard deviation in order to exemplify the achievable precision of the method as if the statistics could have been observed without error.

```
R> obs <- structure(c("T1"=2,"T2"=1),class="simQL")
```

Then, using the default values, we set up the approximation model by

```
R> qsd <- getQLmodel(sim,lb,ub,obs,var.type="wcholMean")
```

and start searching from the parameter `x0` for a root of the QS vector as a crude first estimate of the model parameter.

```
R> QS <- qscoring(qsd, x0=c("mu"=5,"sigma"=3.0))
R> print(QS)

Local method:

 'qscoring'

Start:

    mu     sigma
5.0000    3.0000

Solution:

    mu     sigma
2.2840    1.2883

Quasi-deviance:

 5.456e-08

Iterations.... 5
Status........ 3 ( QFS_STOPVAL_REACHED )

 Optimization stopped because ftol_stop  was reached.

Quasi-score:

[1]    -0.00049572435    0.0013299064
```

The estimated root is already quite satisfactory. However, in practice we would try to improve the current estimate and therfore apply the main estimation function `qle()`. The function alternates between sampling new evaluation points and, in our case, minimizing the quasi-deviance for monitoring the estimation progress. The weights for generating candidate points are automatically adjusted starting at the given weight `0.5`. As before, we test each newly found minimizer whether it could be an approximate root.

```
R> OPT <- qle(qsd,
+         simfunc,
+         nsim=10,
+         global.opts=list("maxeval"=5),
+         local.opts=list("lam_max"=1e-4,"weights"=0.5,
+         "useWeights"=FALSE,"test"=TRUE),cl=cl)
R> print(OPT)

Quasi-deviance:
```

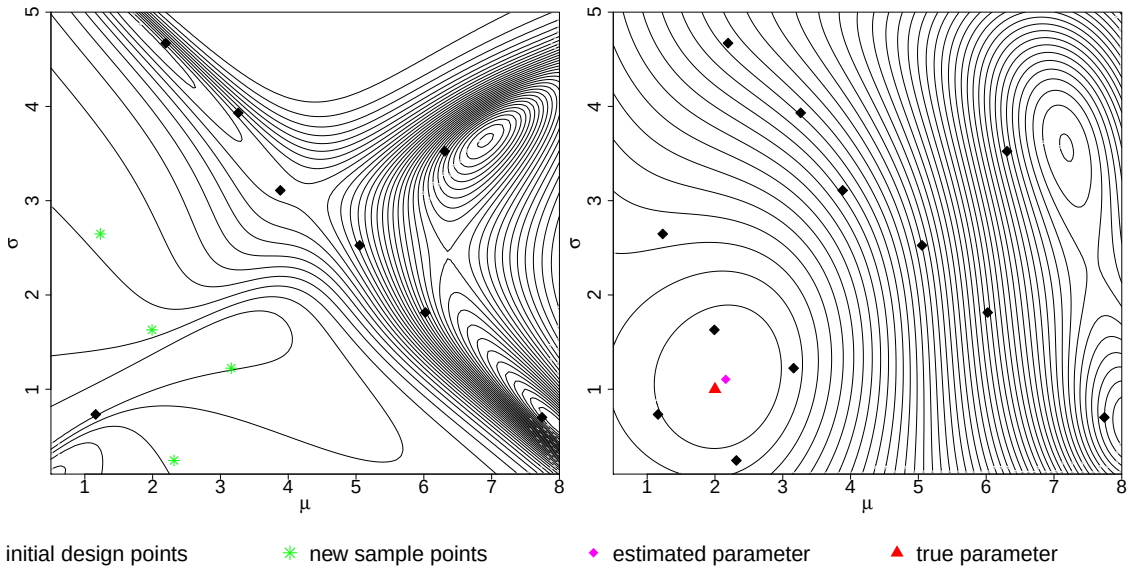

Figure 4: Quasi-devianve level plots: based on the initial design (left) and using additional evaluation points (right).

```
 1.351e-13

Estimate:

    mu     sigma
2.1590    1.1050

Convergence: maximum evaluations reached.

    maxeval        score
5.00000e+00  6.78844e-07

Evaluations:  5  (simulations:  50 )

Variance approximation type:  wcholMean
```

Based on the criteria in (4.25) we can assess the (numerical) consistency of the estimated parameter stored in `OPT$par`

```
R> checkMultRoot(OPT,verbose=TRUE)

       mu sigma minor     value |score_max|     |det|     |max|   |trace|
par 2.159 1.105     0 1.351e-13   6.788e-07 5.204e-05 7.347e-05 2.602e-05
```

where the variable `OPT$qsd` includes the final QL approximation model and can be used for restarts of the algorithm. The above results clearly show the (numerical) convergence of the

current run. Further, the following estimate of the sample variance matrix of the statistics w.r.t. the estimated parameter shows a good match compared to its (weighted) sample average approximation by `Sigma`.

```
R> obs0 <- simQLdata(simfunc,X=OPT$par,nsim=1000,mode="matrix")[[1]]
R> var(obs0)

         T1       T2
T1 0.175810 0.005284
T2 0.005284 0.136526

R> attr(OPT$final,"Sigma")

         T1       T2
T1  0.25224 -0.01539
T2 -0.01539  0.15556

R> stopCluster(cl)
```

5.5. Fitting a Matérn cluster point process model

We fit a stationary Matérn cluster process to the `redwood` point pattern data, see the vignette of package **spatstat** (**?**), once by the classical *method of minimum contrast* and our simulated QL estimation approach. The former estimates the model parameter $\theta = (\kappa, R, \mu)$ of the point process model by minimising the discrepancy between the theoretical K-function $K_\theta(r)$ and its empirical estimate $\hat{K}(r)$ for any radius r over some range $[a, b]$:

$$D(\theta) = \int_a^b \left| K_\theta(r)^q - \hat{K}(r)^q \right|^p \mathrm{d}r \tag{5.3}$$

where $0 \leq a < b$, and $p, q > 0$ are fixed indices. For the latter, we use the empirical intensity $\hat{\lambda}$ (of the whole point process) and $\hat{K}(r)$ with values $r = 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$ as our set of summary statistics.

First, we load the data and fit the model (to the data) by the method of minimum contrast,

```
R> data(redwood)
R> fitMat <- kppm(redwood, ~1, "MatClust")
```

which gives the estimated parameters

```
R> fitMat$modelpar

   kappa        R       mu
24.55790  0.08654  2.52465

R> RNGkind("L'Ecuyer-CMRG")
R> set.seed(297)
```

To initialize the QL estimation method we proceed as before. We define the function which returns the summary statistics:

```
R> simStat <- function(X,cond){
+  x <- Kest(X,r=cond$rr,correction="best")
+  x <- x[[attr(x,"valu")]]
+  x <- x[x>0]
+  if(anyNA(x) || any(!is.finite(x))) {
+    warning(.makeMessage("‘NA‘, ‘NaN‘ or ‘Inf‘ detected.","\n"))
+    x <- x[!is.nan(x) & is.finite(x)]}
+  return(c(intensity(X),x))
+ }
```

as well as the function which first simulates the process model and then returns the values of the employed statistics

```
R> simClust <- function(theta,cond){
+  X <- rMatClust(theta["kappa"],theta["R"],theta["mu"],win=cond$win)
+  simStat(X,cond)
+ }
```

In addition, we set the number of simulation replications, the size of the initial experimental design, the condition object storing the observation window and the fixed values of radii r for the evaluation of $\hat{K}(r)$.

```
R> nsim <- 50
R> Nsample <- 12
R> cond <- list(win=owin(c(0, 2),c(0, 2)),
+               rr=seq(0,0.3,by=0.05))
```

The parameter space is defined by the following lower and upper bound vectors.

```
R> lb <- c("kappa"=20,"R"=0.01,"mu"=1)
R> ub <- c("kappa"=30,"R"=0.25,"mu"=5)
```

We initialize the workers of a local cluster (FORK) which also applies in more general cases of parallel environments in principle.

```
R> cl <- makeCluster(8L)
R> clusterSetRNGStream(cl)
R> clusterCall(cl,fun=function(x) library("spatstat", character.only=TRUE))
R> clusterExport(cl=cl,varlist=c("simStat"), envir=environment())
```

We evaluate our chosen set of summary statistics given the observed point pattern:

```
R> obs0 <- simStat(redwood,cond)
```

and simulate the process model for the first time at randomly generated design points.

```
R> sim <- simQLdata(sim=simClust,cond=cond,nsim=nsim,
+           method="randomLHS",lb=lb,ub=ub,N=Nsample,cl=cl)
```

Putting all together we then construct the QL model.

```
R> qsd <- getQLmodel(sim,lb,ub,obs0,criterion="qle",
+          var.type="kriging",verbose=TRUE)
```

Here we use the kriging approach to approximate the variance matrix estimate of the summary statistics. In general, using the cross-validation (CV) approach for the evaluation of the prediction errors seems to result in more reliable parameter estimates by our method if we use the kriging approach to estimate the variance matrix. Therefore we recommend to analyse the prediction errors of the summary statistics first and, if appropriate, use the CV-based variances instead of those derived from kriging. For this, we first re-fit the leave-one-out cross-validation models of the sample means of the statistics.

```
R> cvm <- prefitCV(qsd, reduce=FALSE, verbose=TRUE)
```

Then we check whether there is any significant bias (e. g. over- or underestimation) in estimating the sample means of the statistics Y by kriging the values at each left-out sample point and averaging over all of them as shown in (4.19).

```
R> crossValTx(qsd, cvm, type = "acve")

     mean.T1  mean.T2   mean.T3   mean.T4    mean.T5  mean.T6  mean.T7
[1,]   1.586 0.001856 0.0009677 -0.001787 -0.0005853 0.003232 0.003799
```

The first statistic (i. e. the empirical intensity) might suffer from systematic overestimation. Therfore we compare the different estimations of each statistic by the their magnitudes of the mean squared CV error:

```
R> crossValTx(qsd, cvm, type = "mse")
```

■echo=FALSE■ matclust$MSEAgain, the first one is more sensitive to left-out sample points of the initial desig

```
R> crossValTx(qsd, cvm, type = "ascve")
\end{Sinput}
\end{Schunk}
<<echo=FALSE>>
matclust$ASCVE
For values close to one the actual error (estimated by the CV approach) would be
equal on average to the error predicted by the corresponding kriging model. In this exampl
above are quite close to zero for each statistic which indicates that the
kriging variances rather overestimate the actual error on average over all
sample points based on the initial design according to the criterion in
\eqref{ASCVE}. However, using the the maximum of both types of prediction errors
ensures that we account for the prediction uncertainty due to the design in a
worst-case scenario while sampling new candidate points and searching for local minimizers
```

```
\begin{Schunk}
\begin{Sinput}
R> attr(cvm,"type") <- "max"
\end{Sinput}
\end{Schunk}
A first run of a global search leads to the following result.
\code{x0}:
\begin{Schunk}
\begin{Sinput}
R> x0 <- c("kappa"=24,"R"=0.08,"mu"=2.5)
R> searchMinimizer(x0,qsd,info=TRUE,
+                  method="direct",cvm=cvm,verbose=TRUE)
\end{Sinput}
\begin{Soutput}
Successful minimization by: direct (status = 5)

Local method:

 'direct'

Start:

   kappa           R          mu
  24.000    0.080000      2.5000

Solution:

  kappa          R         mu
 32.723    0.18798     2.1483

Quasi-deviance:

 2.797e-07

Iterations.... 1000
Status........ 5 ( NLOPT_MAXEVAL_REACHED )

 Optimization stopped because maxeval (above) was reached.

Quasi-score:

[1]    -0.000030417522    0.0032731030       -0.00037689554
\end{Soutput}
\end{Schunk}
We could also apply the quasi-scoring iteration.
\begin{Schunk}
\begin{Sinput}
```

```
R> qscoring(qsd,x0,
+   opts=list("ftol_rel"=1e-6,"slope_tol"=1e-4),
+   cvm=cvm)
\end{Sinput}
\begin{Soutput}
Local method:

 ‘qscoring‘

Start:

   kappa           R          mu
  24.000    0.080000      2.5000

Solution:

   kappa           R          mu
  32.983    0.010877      1.7767

Quasi-deviance:

 0.0002226

Iterations.... 4
Status........ 4 ( QFS_SLOPETOL_REACHED )

 Optimization stopped because slope_tol was reached.

Quasi-score:

[1]    0.0014347213     0.0034586963     -0.0054713035
\end{Soutput}
\end{Schunk}
Here both methods find a solution to the QS vector where the latter ends up in a
local minimizer according to the convergence code. In general, the quasi-scoring
iteration is practically more efficient than the exhaustive global search by the
method \code{direct} (see package \pkg{nloptr}).\par
%
Since the initial approximation of the QS vector results in quite
promising parameter estimates we now start the main estimation routine. We use
the criterion in \eqref{scoreCrit} to select new candidates for evaluation, \code{nextSamp
a maximum of \code{maxeval} evaluations and the quasi-scoring iteration as the
primary local minimization routine or root finding method respectively. In
addition to new evaluation points, we also perform a goodness-of-fit test,
setting \code{test}=\code{TRUE}, of every local minimizer and use
the corresponding simulation results for improving the current kriging
approximations. The following sets some options for the quasi-scoring iteration.
```

```
\begin{Schunk}
\begin{Sinput}
R> qs.opts <-
+  list("xscale"=c(10,0.1,1),
+   "xtol_rel"=1e-10,
+   "ftol_stop"=1e-8,
+   "ftol_rel"=1e-6,
+   "ftol_abs"=1e-4,
+   "score_tol"=1e-4)
\end{Sinput}
\end{Schunk}
The estimation is started by:
\begin{Schunk}
\begin{Sinput}
R> OPT <- qle(qsd, simClust, cond=cond,
+  qscore.opts = qs.opts,
+  global.opts = list("maxiter"=10,"maxeval" = 20,
+                 "weights"=c(50,10,5,1,0.1),
+                 "NmaxQI"=5,"nstart"=100,
+                 "xscale"=c(10,0.1,1)),
+  local.opts = list("lam_max"=1e-2,
+                 "nobs"=200, # number of (bootstrap) observations for testing
+                 "nextSample"="score", # sampling criterion
+                 "ftol_abs"=1e-2, # lower bound on criterion value, triggers testing
+                 "weights"=c(0.55), # constant weight factor
+                 "eta"=c(0.025,0.075), # ignored, automatic adjustment of weights
+                 "test"=TRUE), # testing approximate root is enabled
+                 method = c("qscoring","bobyqa","direct"), # restart methods
+                 errType="max", # use max of kriging and CV error
+         iseed=297, cl=cl) # store seed and use given cluster object
\end{Sinput}
\end{Schunk}
The result of the quasi-likelihood estimation is
\begin{Schunk}
\begin{Sinput}
R> print(OPT)
\end{Sinput}
\begin{Soutput}
Quasi-deviance:

 2.414e-07

Estimate:

   kappa           R          mu
  21.394    0.068207      2.9090
```

```
Convergence: maximum evaluations reached.

    maxeval
2.00000e+01

Evaluations:  20  (simulations:  1000 )

Variance approximation type:  kriging
\end{Soutput}
\end{Schunk}
```

where we also can extract further information of the optimization results by

```
\begin{Schunk}
\begin{Sinput}
R> attr(OPT,"optInfo")
\end{Sinput}
\begin{Soutput}
$x0
kappa     R    mu
27.50  0.13  3.00

$W
NULL

$theta
NULL

$last.global
[1] FALSE

$minimized
[1] TRUE

$useCV
[1] TRUE

$method
[1] "qscoring"

$nsim
[1] 50

$iseed
[1] 297
\end{Soutput}
\end{Schunk}
```

The matrix \code{W} is the quasi-information matrix at the parameter \code{theta} as the next to the final parameter estimate. We can also inspect

```
the final minimization results
\begin{Schunk}
\begin{Sinput}
R> OPT$final
\end{Sinput}
\begin{Soutput}
Local method:

 'qscoring'

Start:

   kappa           R          mu
  22.591    0.067941      2.7655

Solution:

   kappa           R          mu
  21.394    0.068207      2.9090

Quasi-deviance:

 2.414e-07

Iterations.... 19
Status........ 6 ( QFS_STEPMIN_REACHED )

 Optimization stopped because minimum relative direction length was reached.

Quasi-score:

[1]    0.00016922149    -0.0056384810    0.0014980617
\end{Soutput}
\end{Schunk}
and, based on the above results, we use the final quasi-deviance approximation
(and hence QS approximation) to search for other local minimizers.
\begin{Schunk}
\begin{Sinput}
R> S0 <- searchMinimizer(OPT$par,OPT$qsd,
+        method="bobyqa",cvm=OPT$cvm,verbose=TRUE)
\end{Sinput}
\begin{Soutput}
Successful minimization by: bobyqa (status = 1)
\end{Soutput}
\end{Schunk}
If the user is unsatisfied with the estimation result, we can easily restart
the main estimation routine \code{qle} or start with a local quasi-scoring
```

```
iteration from the last resulting parameter without simulating the model.
\begin{Schunk}
\begin{Sinput}
R> QS <- qscoring(OPT$qsd,OPT$par,
+        opts=list("slope_tol"=1e-4,"score_tol"=1e-3),
+        cvm=OPT$cvm)
\end{Sinput}
\end{Schunk}
The additionally estimated parameters have, however, not much improved as shown
by the following consistency checks.
\begin{Schunk}
\begin{Sinput}
R> par <- rbind("QS"=QS$par,"S0"=S0$par)
R> checkMultRoot(OPT,par=par)
\end{Sinput}
\begin{Soutput}
     kappa       R    mu minor     value |score_max|   |det|  |max| |trace|
par  21.39 0.06821 2.909     2 2.316e-06     0.03211  49.939  7.468   2.330
QS * 21.40 0.06819 2.908     3 1.146e-06     0.03469   3.848  6.831   2.698
S0   21.39 0.06820 2.909     2 1.673e-06     0.03081 125.165 19.040   8.586
\end{Soutput}
\end{Schunk}
which shows by the row with an asterisk that the corresponding parameter
\begin{Schunk}
\begin{Sinput}
R> QS$par
\end{Sinput}
\begin{Soutput}
   kappa        R       mu
21.40181  0.06819  2.90796
\end{Soutput}
\end{Schunk}
is the best one according to the criteria defined in Section \ref{subsec:check}.
We can assess the goodness-of-fit (see Section \ref{subsec:testing}) of
the estimated parameter based on a simulated sample of size \code{nsim}.
\begin{Schunk}
\begin{Sinput}
R> par0 <- OPT$par
R> obs0 <- OPT$qsd$obs
R> # testing ‘par0‘ with observed statistics ‘obs0‘
R> # which can be replaced by the user and are obsolete below
R> Stest <- qleTest(OPT, # estimation results
+           par0=par0, # parameter to test
+           obs0=obs0, # alternative observed statistics
+           sim=simClust,cond=cond,nsim=100,
+           method=c("qscoring","bobyqa","direct"), # restart methods
+           opts=qs.opts,control=list("ftol_abs"=1e-8), # minimization options
```

```
+           multi.start=1L,cl=cl,verbose=TRUE) # multi-start and parallel options
\end{Sinput}
\end{Schunk}
\begin{Schunk}
\begin{Sinput}
R> print(Stest)
\end{Sinput}
\begin{Soutput}
Call:

qleTest(OPT, par0 = par0, obs0 = obs0, sim = simClust, cond = cond,
    nsim = 100, method = c("qscoring", "bobyqa", "direct"), opts = qs.opts,
    control = list(ftol_abs = 1e-08), multi.start = 1L, cl = cl,
    verbose = TRUE)

Coefficients:

       Estimate   Std. Error  RMSE       Bias       Mean
kappa  21.394313  3.2843479   3.5499350  1.3867108  22.781024
R       0.068207  0.0094823   0.0096558  0.0020543   0.070261
mu      2.909003  0.4993318   0.4968991  0.0083545   2.917357


Bootstrap Score-test:

   s value      Pr(>s)
2.4143e-07     0.44554


Average quasi-score:

[1]  -0.040720   0.922214  -0.019203


Predicted Std. Errors:

       Average   Estimate  EF
kappa  6.432210  5.928181  0.66994
R      0.022549  0.017701  0.83318
mu     0.929723  0.836024  0.68248
\end{Soutput}
\end{Schunk}
and stop the cluster:
\begin{Schunk}
\begin{Sinput}
R> stopCluster(cl)
\end{Sinput}
```

```
\end{Schunk}
The results show that the simulated data set provides not enough evidence
to reject the null hypothesis and thus we cannot distinguish the observed
statistics from the simulated output of the model. The last column shows the
relative difference of the empirical and predicted error at the estimated
parameter as defined in \eqref{relDif}.\par
%
Finally, a short error analysis shows that the predicted standard error
\begin{Schunk}
\begin{Sinput}
R> diag(attr(Stest,"qi"))^0.5
\end{Sinput}
\begin{Soutput}
 kappa      R     mu
5.9282 0.0177 0.8360
\end{Soutput}
\end{Schunk}
of the final parameter (see column \code{Estimate}) and its empirical error
\begin{Schunk}
\begin{Sinput}
R> sqrt(diag(attr(Stest,"msem")))
\end{Sinput}
\begin{Soutput}
   kappa        R       mu
3.549935 0.009656 0.496899
\end{Soutput}
\end{Schunk}
which is the square root of the diagonal terms of the \emph{mean squared error
matrix} of the re-estimated parameters
\begin{Schunk}
\begin{Sinput}
R> attr(Stest,"msem")
\end{Sinput}
\begin{Soutput}
          kappa         R        mu
kappa 12.602038 3.837e-03 -0.895481
R      0.003837 9.323e-05  0.001454
mu    -0.895481 1.454e-03  0.246909
\end{Soutput}
\end{Schunk}
have the same order of magnitude and compare well with each other.\par
%
\begin{figure}[ht!]
\centering
  \includegraphics{Kfunc.pdf}
  \caption{Pointwise envelopes of the summary statistics $\hat{K}(r)$, $\hat{G}(r)$ and $\
  based on simulated patterns of two fitted point process models for the
```

```
  \code{redwood} point pattern data: fitted by the method of minimum contrast
  (left column) and quasi-likelihood (right column).}
\label{fig:matclust}
\end{figure}
%
Finally, we compare the simulation envelopes based on the summary statistics
$\hat{K}$ and additionally $\hat{G}$ and $\hat{F}$. The plot in Figure
\ref{fig:matclust} suggests a good agreement between the model and the data for
both estimation methods with slightly better results by the quasi-likelihood
estimation approach regarding the envelopes of $\hat{G}$ and $\hat{F}$.
%%
\section{Conclusion}\label{sec:conclusion}
The package \pkg{qle} provides methods for parameter estimation for
a generic class of parametric statistical models. The estimation
approach is entirely simulation-based, in that moment characteristics of the
unknown distributions of the statistics are infered from model replications and
approximated by a specific kriging approach. Therefore, as a preliminary step,
the construction of the kriging approximations requires an experimental design
of randomly generated parameters or points. Also, users can assess its
predictive quality by error measures specifically tailored to the needs of the QL
estimation approach.\par
%
The estimation is based on finding a root of the QS vector but also subsumes
other estimation methods, e.\,g. simulated method of moments or least squares,
based on the Mahalanobis distance. Optionally, we could search for a suitable
starting parameter for our QL estimation approach by one of these methods first
because the corresponding criterion is potentially easier to compute and
minimize, particularly for small sample sizes, over the parameter space. Also we
can use gradient based and derivative-free methods, see package \pkg{nloptr}
\citep{pkg:nloptr}, for minimization of both criterion functions. On the
contrary, the quasi-scoring algorithm is solely used for root finding of the QS vector
in conjunction with the QD as a monitor function. Although the latter might
suffer from numerical instabilities due to sparsely sampled regions of the parameter space
(especially in the beginning of the estimation procedure in which case other
methods are employed automatically) it usually needs only a fraction of iterations compare
to other general purpose solvers.\par
%
In practice, it might be especially advantageous to (temporarily)
store or cache computational results of functions for a deeper analysis of
estimation results or even errors. Setting \code{options("qle.cache"=TRUE)}
persistently stores (and reloads) any results of the following functions:
\begin{itemize}
 \item[] \code{simQLdata}, \code{prefitCV}, \code{mahalDist}, \code{quasiDeviance},
 \item[] \code{fitCov}, \code{fitSIRFk}, \code{updateCovModels}, \code{getQLmodel}
\end{itemize}
using package \pkg{digest} \citep{pkg:digest} for hash digests of \proglang{R}
objects to create file names. Further, several options are available for
```

```
estimating the variance matrix of the statistics (e.\,g. using package \pkg{expm} \citep{p
for matrix logarithm transformations) as well as two types of prediction
variances (of sample mean values of statistics). These are used mainly in order to account
simulation error of the statistics while searching for potential candidates of the
unknown model parameter (either sampling from a multivariate normal based on the
package \pkg{mvtnorm} \citep{pkg:mvtnorm} or uniform distribution).
Finally, the user can perform a Monte Carlo hypothesis test in order to assess
the goodness-of-fit of the parametric model which does not involve additional simulations
(except for generating ''observations`` according to the fitted statistical model) and pro
predicted and empirically estimated standard errors of the model parameter.\par
%
\section{Computational details}
The package \pkg{qle} is implemented in \proglang{R} with extensions written in
\proglang{C/C++}. It can be found on the Comprehensive \proglang{R} Archive
Network at (CRAN, \url{http://CRAN.R-project.org/package=qle}) and also is hosted on
R-Forge at (\url{http://r-forge.r-project.org/projects/qle}). It uses a
NAMESPACE and depends on the package \pkg{parallel} \citep{pkg:stats} and
\pkg{nloptr} \citep{pkg:nloptr} among others (see above). To reproduce the
examples of the vignette, which are also provided as separate \proglang{R}
source files, we recommend the package \pkg{spatstat}. The package \pkg{qle}
automatically compiles when installed.
%
\section*{Acknowledgments}
The author would like to thank the German Science Foundation (DFG) for financial support o
this research within the framework of the priority program ''Life$\infty$``
(SPP 1466).
%
```

Affiliation:

Markus Baaske
Faculty of Mathematics and Computer Science
Friedrich Schiller University Jena
07743 Jena, Germany
Email: markus.baaske@uni-jena.de

```
\relax
\providecommand\hyper@newdestlabel[2]{}
\bibstyle{jss}
\providecommand\HyperFirstAtBeginDocument{\AtBeginDocument}
\HyperFirstAtBeginDocument{\ifx\hyper@anchor\@undefined
\global\let\oldcontentsline\contentsline
\gdef\contentsline#1#2#3#4{\oldcontentsline{#1}{#2}{#3}}
\global\let\oldnewlabel\newlabel
\gdef\newlabel#1#2{\newlabelxx{#1}#2}
\gdef\newlabelxx#1#2#3#4#5#6{\oldnewlabel{#1}{{#2}{#3}}}
\AtEndDocument{\ifx\hyper@anchor\@undefined
\let\contentsline\oldcontentsline
\let\newlabel\oldnewlabel
\fi}
\fi}
\global\let\hyper@last\relax
\gdef\HyperFirstAtBeginDocument#1{#1}
\providecommand\HyField@AuxAddToFields[1]{}
\providecommand\HyField@AuxAddToCoFields[2]{}
\citation{Baaske2014}
\citation{Baaske2018}
\citation{Godambe1991}
\citation{Jesus871}
\citation{ref:Heyde1997}
\citation{ref:Myers1995}
\citation{ref:Sacksb1989,ref:Cressie1993,ref:Kleijnen2009}
\citation{ref:Jones1998}
\citation{ref:Jones2001}
\citation{pkg:stats}
\citation{ref:Wedder1974}
\citation{ref:Heyde1997}
\citation{pkg:gee}
\citation{ref:Liang1986}
\citation{ref:pkgGMM}
\citation{ref:Hansen1982}
\citation{ref:McFadden1989}
\citation{pkg:spatstat}
\citation{ref:Heyde1997}
```

```
\citation{ref:Liang1995}
\newlabel{sec:QL}{{2}{4}{}{section.2}{}}
\newlabel{score}{{2.1}{4}{}{equation.2.1}{}}
\newlabel{information}{{2.3}{4}{}{equation.2.3}{}}
\citation{Dryden2009}
\newlabel{qi}{{2.4}{5}{}{equation.2.4}{}}
\newlabel{sec:SQLE}{{3}{5}{}{section.3}{}}
\newlabel{approxscore}{{3.1}{5}{}{equation.3.1}{}}
\newlabel{QI}{{3.2}{5}{}{equation.3.2}{}}
\newlabel{approxQD}{{3.3}{6}{}{equation.3.3}{}}
\@writefile{lof}{\contentsline {figure}{\numberline {1}{\ignorespaces Algorithmic
strucure of function \bgroup \catcode `\_12\relax \catcode `\~12\relax \catcode
`\$12\relax {\normalfont  \ttfamily  \hyphenchar \font =-1 qle}\egroup \relax }}{7}{figure
\providecommand*\caption@xref[2]{\@setref\relax\@undefined{#1}}
\newlabel{fig:alg}{{1}{7}{Algorithmic strucure of function \code {qle}\relax }{figure.capt
\citation{ref:Osborne1992}
\citation{ref:Conn2009}
\newlabel{subsec:qsSolve}{{3.2}{8}{}{subsection.3.2}{}}
\newlabel{QSE}{{3.6}{8}{}{equation.3.6}{}}
\newlabel{qscoring}{{3.8}{8}{}{equation.3.8}{}}
\citation{ref:Golub1996}
\newlabel{eOP}{{3.9}{9}{}{equation.3.9}{}}
\newlabel{simple}{{3.11}{9}{}{equation.3.11}{}}
\newlabel{qsErr}{{3.3}{9}{}{subsection.3.3}{}}
\newlabel{qsError}{{3.13}{9}{}{equation.3.13}{}}
\citation{Baaske2014}
\newlabel{subsec:termCond}{{3.4}{10}{}{subsection.3.4}{}}
\newlabel{sec:extent}{{4}{10}{}{section.4}{}}
\newlabel{subsec:local}{{4.1}{10}{}{subsection.4.1}{}}
\newlabel{SS}{{4.1}{10}{}{equation.4.1}{}}
\citation{ref:Regis2007}
\newlabel{stilde}{{4.3}{11}{}{equation.4.3}{}}
\newlabel{scoreCrit}{{4.4}{11}{}{equation.4.4}{}}
\citation{ref:Puk2006}
\newlabel{subsec:global}{{4.2}{12}{}{subsection.4.2}{}}
\citation{ref:Jakobsson2010}
\citation{Baaske2014}
\citation{Dryden2009}
```

\newlabel{globsamp}{{4.6}{13}{}{equation.4.6}{}}
\newlabel{subsec:varianceInter}{{4.3}{13}{}{subsection.4.3}{}}
\citation{ref:Golub1996}
\citation{Dryden2009}
\citation{Qi2007}
\citation{ref:Heyde1997}
\citation{Baaske2014}
\newlabel{L}{{4.7}{14}{}{equation.4.7}{}}
\newlabel{VC}{{4.8}{14}{}{equation.4.8}{}}
\newlabel{VL}{{4.9}{14}{}{equation.4.9}{}}
\newlabel{wVC}{{4.10}{14}{}{equation.4.10}{}}
\newlabel{kernel}{{4.11}{14}{}{equation.4.11}{}}
\newlabel{wVL}{{4.12}{14}{}{equation.4.12}{}}
\newlabel{modVar}{{4.13}{14}{}{equation.4.13}{}}
\citation{ref:ChilesDelfiner1999}
\citation{ref:Marchant2007}
\citation{Kleijnen2004a}
\citation{Jin2002}
\citation{Kleijnen2004a}
\newlabel{subsec:altVar}{{4.4}{15}{}{subsection.4.4}{}}
\newlabel{pseudo}{{4.15}{15}{}{equation.4.15}{}}
\citation{Shao1995}
\citation{Meckesheimer2002}
\newlabel{jackVar}{{4.16}{16}{}{equation.4.16}{}}
\newlabel{rmsecv}{{4.17}{16}{}{equation.4.17}{}}
\newlabel{pseudonc}{{4.18}{16}{}{equation.4.18}{}}
\newlabel{sec:design}{{4.5}{16}{}{subsection.4.5}{}}
\citation{ref:Sacksb1989}
\citation{mueller2001}
\citation{ref:Sacksa1989}
\citation{ref:Wackernagel2003}
\citation{ref:ChilesDelfiner1999}
\citation{ref:Wackernagel2003}
\newlabel{ACVE}{{4.19}{17}{}{equation.4.19}{}}
\newlabel{MSE}{{4.20}{17}{}{equation.4.20}{}}
\citation{pkg:lhs}
\citation{ref:Heyde1997}
\citation{ref:Heyde1997}

\citation{ref:Heyde1997}
\newlabel{SCVE}{{4.21}{18}{}{equation.4.21}{}}
\newlabel{ASCVE}{{4.22}{18}{}{equation.4.22}{}}
\newlabel{subsec:check}{{4.6}{18}{}{subsection.4.6}{}}
\newlabel{Qid}{{4.23}{18}{}{equation.4.23}{}}
\citation{Ripley2009}
\citation{ref:Heyde1997}
\citation{ref:Heyde1997}
\citation{ref:Heyde1997}
\newlabel{termCond}{{{4.25}}{19}{}{equation.4.24}{}}
\newlabel{subsec:testing}{{4.7}{19}{}{subsection.4.7}{}}
\citation{efron1994}
\newlabel{empQuantile}{{4.26}{21}{}{equation.4.26}{}}
\newlabel{relDif}{{4.27}{21}{}{equation.4.27}{}}
\newlabel{sec:Restim}{{5}{21}{}{section.5}{}}
\citation{Beers2003}
\newlabel{subsec:parallel}{{5.2}{23}{}{subsection.5.2}{}}
\newlabel{subsec:mmq}{{5.3}{23}{}{subsection.5.3}{}}
\@writefile{lof}{\contentsline {figure}{\numberline {2}{\ignorespaces M/M/1 queue: number of customers (left) and quasi-deviance, respectively, quasi-score approximation (right). All approximations shown are based on the same set of evaluation points.\relax }}{26}{figure.caption.2}}
\newlabel{fig:mm1q1}{{2}{26}{M/M/1 queue: number of customers (left) and quasi-deviance, respectively, quasi-score approximation (right). All approximations shown are based on the same set of evaluation points.\relax }{figure.caption.2}{}}
\@writefile{lof}{\contentsline {figure}{\numberline {3}{\ignorespaces Final quasi-deviance and quasi-score function approximation after adding five new evaluation points.\relax }}{27}{figure.caption.3}}
\newlabel{fig:mm1q2}{{3}{27}{Final quasi-deviance and quasi-score function approximation after adding five new evaluation points.\relax }{figure.caption.3}{}}
\newlabel{rhomle}{{5.1}{27}{}{equation.5.1}{}}
\newlabel{qsmm1q}{{5.2}{27}{}{equation.5.2}{}}
\@writefile{lot}{\contentsline {table}{\numberline {1}{\ignorespaces Empirical mean squared error (MSE) and average variance $\overline {Var}$ of the re-estimated parameters for 100 observations using the quasi-likelihood approach and maximum likelihood method.\relax }}{29}{table.caption.4}}
\newlabel{tab:err}{{1}{29}{Empirical mean squared error (MSE) and average variance $\overline {Var}$ of the re-estimated parameters for 100 observations using the quasi-likelihood approach and maximum likelihood method.\relax }{table.caption.4}{}}
\@writefile{lof}{\contentsline {figure}{\numberline {4}{\ignorespaces Quasi-devianve level plots: based on the initial design (left) and using additional evaluation