Statistics and Computing (2021) 31:38
https://doi.org/10.1007/s11222-021-10012-y

A robust and efficient algorithm to find profile likelihood confidence intervals

Samuel M. Fischer[1,2] · Mark A. Lewis[2,3]

Received: 31 March 2020 / Accepted: 9 January 2021 / Published online: 7 May 2021

Abstract

Profile likelihood confidence intervals are a robust alternative to Wald's method if the asymptotic properties of the maximum likelihood estimator are not met. However, the constrained optimization problem defining profile likelihood confidence intervals can be difficult to solve in these situations, because the likelihood function may exhibit unfavorable properties. As a result, existing methods may be inefficient and yield misleading results. In this paper, we address this problem by computing profile likelihood confidence intervals via a trust-region approach, where steps computed based on local approximations are constrained to regions where these approximations are sufficiently precise. As our algorithm also accounts for numerical issues arising if the likelihood function is strongly non-linear or parameters are not estimable, the method is applicable in many scenarios where earlier approaches are shown to be unreliable. To demonstrate its potential in applications, we apply our algorithm to benchmark problems and compare it with 6 existing approaches to compute profile likelihood confidence intervals. Our algorithm consistently achieved higher success rates than any competitor while also being among the quickest methods. As our algorithm can be applied to compute both confidence intervals of parameters and model predictions, it is useful in a wide range of scenarios.

Keywords Computer algorithm · Constrained optimization · Parameter estimation · Estimability · Identifiability

1 Introduction

1.1 Profile likelihood confidence intervals

Confidence intervals are an important tool for statistical inference, used not only to assess the range of predictions that are supported by a model and data but also to detect potential estimability issues (Raue et al. 2009). These estimability issues occur if not enough data are available to infer a statistical parameter on the desired confidence level, and the corresponding confidence intervals are infinite (Raue et al. 2009). Due to the broad range of applications, confidence intervals are an integral part of statistical model analysis and widely used across disciplines.

Often, confidence intervals are constructed via Wald's method, which exploits the asymptotic normality of the maximum likelihood estimator (MLE). Though Wald's method is accurate in "benign" use cases, the approach can be imprecise or fail if not enough data are available to reach the asymptotic properties of the MLE. This will be the case, in particular, if the MLE is not unique, i.e. parameters are not identifiable, or if the likelihood is very sensitive to parameter changes beyond some threshold, e.g. in dynamical systems undergoing bifurcations (see e.g. Ramsay et al. 2007). Therefore, other methods, such as profile likelihood techniques (Cox and Snell 1989), are favorable in many use cases.

Both Wald-type and profile likelihood confidence intervals are constructed by inverting the likelihood ratio test. That is, the confidence interval for a parameter θ_0 encompasses all values $\bar{\theta}_0$ that might suit as acceptable null hypotheses if the

The primary work for this article was conducted at the University of Alberta.

✉ Samuel M. Fischer
samuel.fischer@ufz.de

1 UFZ - Helmholtz Centre for Environmental Research, Department of Ecological Modelling, Permoserstraße 15, 04318 Leipzig, Germany

2 Department of Mathematical and Statistical Sciences, University of Alberta, 632 Central Academic Building, Edmonton, AB T6G 2G1, Canada

3 Department of Biological Sciences, University of Alberta, CW 405, Biological Sciences Building, Edmonton, AB T6G 2E9, Canada

parameter were to be fixed; i.e. $H_0\colon \theta_0 = \bar{\theta}_0$ could not be rejected versus the alternative $H_1\colon \theta_0 \neq \bar{\theta}_0$. As the likelihood ratio statistic is, under regularity conditions, approximately χ^2 distributed under the null hypothesis, the confidence interval is given by

$$I = \left[\bar{\theta}_0 \,\middle|\, 2\left(\max_{\boldsymbol{\theta}\in\Theta} \ell(\boldsymbol{\theta}) - \max_{\boldsymbol{\theta}\in\Theta\colon \theta_0=\bar{\theta}_0} \ell(\boldsymbol{\theta})\right) \leq \chi^2_{1,1-\alpha}\right], \tag{1}$$

where $\boldsymbol{\theta} \in (\theta_0, \ldots, \theta_{n-1})^\top$ is a parameter vector in the parameter space Θ, ℓ denotes the log-likelihood function, α is the desired confidence level, and $\chi^2_{k,1-\alpha}$ is the $(1-\alpha)$th quantile of the χ^2 distribution with k degrees of freedom. We assume without loss of generality that we seek a confidence interval for the first entry of the parameter vector.

The function that maps $\bar{\theta}_0$ to the constrained maximum

$$\ell_{\mathrm{PL}}\left(\bar{\theta}_0\right) := \max_{\boldsymbol{\theta}\in\Theta\,:\,\theta_0=\bar{\theta}_0} \ell(\boldsymbol{\theta}) \tag{2}$$

is called the profile log-likelihood. While Wald's method approximates ℓ and ℓ_{PL} as quadratic functions, profile likelihood confidence intervals are constructed by exact computation of the profile log-likelihood ℓ_{PL}. This makes this method more accurate but also computationally challenging.

1.2 Existing approaches

Conceptually, the task of identifying the end points $\theta_0^{\min}$ and $\theta_0^{\max}$ of the confidence interval I is equivalent to finding the maximal (or minimal) value for θ_0 with

$$\ell_{\mathrm{PL}}(\theta_0) = \ell^* := \ell\left(\hat{\boldsymbol{\theta}}\right) - \frac{1}{2}\chi^2_{1,1-\alpha}, \tag{3}$$

Here, $\hat{\boldsymbol{\theta}}$ denotes the MLE; the value ℓ^* follows from rearranging the terms in the inequality characterizing I [see Eq. (1)].

There are two major perspectives to address this problem. It could either be understood as a one-dimensional root finding problem on ℓ_{PL} or as the constrained maximization (or minimization) problem

$$\theta_0^{\max} = \max_{\boldsymbol{\theta}\in\Theta\colon \ell(\boldsymbol{\theta})\geq\ell^*} \theta_0 \tag{4}$$

($\theta_0^{\min}$ analog). Approaches developed from either perspective face the challenge of balancing efficiency against robustness, the ability to return correct results even for difficult problems.

The root finding perspective (Cook and Weisberg 1990; DiCiccio and Tibshirani 1991; Stryhn and Christensen 2003; Moerbeek et al. 2004; Ren and Xia 2019) is robust if small steps are taken and solutions of the maximization problem (2) are good initial guesses for the maximizations in later steps. Nonetheless, the step size should be variable if the confidence intervals might be large or parameters might be inestimable, i.e. the profile log-likelihood ℓ_{PL} never falls below the threshold ℓ^*. At the same time, care must be taken with large steps, as solving (2) can be difficult if the initial guesses are poor, and algorithms may fail to converge. Therefore, conservative step choices are often advisable even though they may decrease the overall efficiency of the approaches.

The constrained maximization perspective (Neale and Miller 1997; Wu and Neale 2012) has the advantage that efficient solvers for such problems are readily implemented in many optimization packages. If the likelihood function is "well behaved", these methods converge very quickly. However, in practical problems, the likelihood function may have local extrema, e.g. due to lack of data, or steep "cliffs" that may hinder these algorithms from converging to a feasible solution. Furthermore, general algorithms are typically not optimized for problems like (4), in which the target function is simple and the major challenge is in ensuring that the constraint is met. Therefore, an approach would be desirable that is specifically tailored to solve the constrained maximization (4) in a robust and efficient manner.

A first step in this direction is the algorithm by Venzon and Moolgavkar (1988), which solves (4) by repeated quadratic approximations of the likelihood surface. As the method is of Newton–Raphson type, it is very efficient as long as the local approximations are accurate. Therefore, the algorithm is fast if enough data are available to make the considered model locally approximately normal. Otherwise, the algorithm relies heavily on good initial guesses. Though methods to determine accurate initial guesses exist (Gimenez et al. 2005), the algorithm by Venzon and Moolgavkar (1988) (below abbreviated as VM) can get stuck in local extrema or fail to converge if the likelihood surface is non-convex, has sudden jumps, or other unfavorable properties (see e.g. Ren and Xia 2019). Moreover, the algorithm will break down if parameters are not identifiable. Thus, VM cannot be applied in some important use cases of profile likelihood confidence intervals.

1.3 Our contributions

In this paper, we address the issues of VM by introducing an algorithm extending the ideas of Venzon and Moolgavkar (1988). Our algorithm, which we will call *Robust Venzon–Moolgavkar Algorithm* (RVM) below combines the original procedure with a trust region approach (Conn et al. 2000; Yuan 2015). That is, the algorithm never steps outside of the region in which the likelihood approximation is sufficiently precise. Furthermore, RVM accounts for unidentifiable parameters, local minima and maxima, and sharp changes in the likelihood surface. The algorithm is implemented in the Python package *ci-rvm* that can be retrieved from the Python package index (see pypi.org/project/ci-rvm).

The approach by Venzon and Moolgavkar (1988) has been criticized for not being directly applicable to construct confidence intervals for functions of parameters (Pek and Wu 2015). Often the main research interest is not in identifying specific model parameters but in obtaining model predictions, which can be expressed as a function of the parameters. We show how RVM—and other methods, including VM—can also be applied to determine confidence intervals for functions of parameters.

This paper is structured as follows: in the first section, we start by outlining the main ideas behind RVM before we provide details of the applied procedures. Furthermore, we briefly describe how the algorithm can be used to determine confidence intervals of functions of parameters. In the second section, we apply RVM and alternative algorithms to a model fitting problem with empirical data and a range of benchmark problems with simulated data. We conclude this paper with a discussion of the test results and the benefits and limitations of RVM in comparison to earlier methods.

2 Algorithm

2.1 Basic ideas

Suppose we consider a model with an n-dimensional parameter vector $\boldsymbol{\theta} := (\theta_0, \dots, \theta_{n-1})^\top$ and a twice continuously differentiable log-likelihood function ℓ. Assume without loss of generality that we seek to construct a level-α confidence interval for the parameter θ_0, and let $\widetilde{\boldsymbol{\theta}} := (\theta_1, \dots, \theta_{n-1})^\top$ be the vector of all remaining parameters, called nuisance parameters. For convenience, we may write $\ell = \ell(\boldsymbol{\theta})$ as a function of the complete parameter vector or $\ell = \ell\left(\theta_0, \widetilde{\boldsymbol{\theta}}\right)$ as a function of the parameter of interest and the nuisance parameters.

The algorithm RVM introduced in this paper searches the right end point $\theta_0^{\max}$ (equation (4)) of the confidence interval I. The left end point can be identified with the same approach if a modified model is considered in which ℓ is flipped in θ_0. As RVM builds on the method by Venzon and Moolgavkar (1988), we start by recapitulating their algorithm VM below.

Let $\boldsymbol{\theta}^* \in \Theta$ be the parameter vector at which the parameter of interest is maximal, $\theta_0^* = \theta_0^{\max}$, and $\ell\left(\boldsymbol{\theta}^*\right) \geq \ell^*$. Venzon and Moolgavkar (1988) note that $\boldsymbol{\theta}^*$ satisfies the following necessary conditions:

1. $\ell\left(\boldsymbol{\theta}^*\right) = \ell^*$ and
2. ℓ is in a local maximum with respect to the nuisance parameters, which implies $\frac{\partial \ell}{\partial \widetilde{\boldsymbol{\theta}}}(\boldsymbol{\theta}^*) = 0$.

The algorithm VM searches for $\boldsymbol{\theta}^*$ by minimizing both the log-likelihood distance to the threshold $|\ell(\boldsymbol{\theta}) - \ell^*|$ and the magnitude of the gradient of the nuisance parameters $|\frac{\partial \ell}{\partial \widetilde{\boldsymbol{\theta}}}|$. To this end, the algorithm repeatedly approximates the log-likelihood surface ℓ with second order Taylor expansions $\hat{\ell}$. If $\boldsymbol{\theta}^{(i)}$ is the parameter vector in the ith iteration of the algorithm, expanding ℓ around $\boldsymbol{\theta}^{(i)}$ yields

$$\begin{aligned}\hat{\ell}(\boldsymbol{\theta}) &:= \ell\left(\boldsymbol{\theta}^{(i)}\right) + \boldsymbol{g}^\top\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}\right) \\ &\quad + \frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}\right)^\top \underline{\mathbf{H}}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}\right) \\ &= \bar{\ell} + \widetilde{\boldsymbol{g}}^\top\widetilde{\boldsymbol{\delta}} + g_0\delta_0 + \frac{1}{2}\widetilde{\boldsymbol{\delta}}^\top\underline{\widetilde{\mathbf{H}}}\widetilde{\boldsymbol{\delta}} + \delta_0\widetilde{\mathbf{H}}_0^\top\widetilde{\boldsymbol{\delta}} + \frac{1}{2}\delta_0 \mathrm{H}_{00}\delta_0 \\ &=: \hat{\ell}^\delta\left(\delta_0, \widetilde{\boldsymbol{\delta}}\right). \end{aligned} \tag{5}$$

Here, $\boldsymbol{\delta} := \boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}$, $\bar{\ell} := \ell\left(\boldsymbol{\theta}^{(i)}\right)$; $\boldsymbol{g} := \frac{\partial \ell}{\partial \boldsymbol{\theta}}\left(\boldsymbol{\theta}^{(i)}\right)$ is the gradient and $\underline{\mathbf{H}} := \frac{\partial^2 \ell}{\partial \boldsymbol{\theta}^2}\left(\boldsymbol{\theta}^{(i)}\right)$ the Hessian matrix of ℓ at $\boldsymbol{\theta}^{(i)}$. Analogously to notation used above, we split $\boldsymbol{\delta}$ into its first entry δ_0 and the remainder $\widetilde{\boldsymbol{\delta}}$, $\boldsymbol{g}$ into g_0 and $\widetilde{\boldsymbol{g}}$, and write $\mathbf{H}_0$ for the first column of $\underline{\mathbf{H}}$, $\underline{\widetilde{\mathbf{H}}}$ for $\underline{\mathbf{H}}$ without its first column and row, and split $\mathbf{H}_0$ into H_{00} and $\widetilde{\mathbf{H}}_0$.

In each iteration, VM seeks δ_0^* and $\widetilde{\boldsymbol{\delta}}^*$ that satisfy conditions 1 and 2. Applying condition 2 to the approximation $\hat{\ell}^\delta$ [Eq. (5)] yields

$$\widetilde{\boldsymbol{\delta}}^* = -\underline{\widetilde{\mathbf{H}}}^{-1}\left(\widetilde{\mathbf{H}}_0\delta_0 + \widetilde{\boldsymbol{g}}\right). \tag{6}$$

Inserting (5) and (6) into condition 1 gives us

$$\begin{aligned}\ell^* &= \frac{1}{2}\left(\mathrm{H}_{00} - \widetilde{\mathbf{H}}_0^\top\underline{\widetilde{\mathbf{H}}}^{-1}\widetilde{\mathbf{H}}_0\right)\delta_0^{*2} \\ &\quad + \left(g_0 - \widetilde{\boldsymbol{g}}^\top\underline{\widetilde{\mathbf{H}}}^{-1}\widetilde{\mathbf{H}}_0\right)\delta_0^* + \bar{\ell} - \frac{1}{2}\widetilde{\boldsymbol{g}}^\top\underline{\widetilde{\mathbf{H}}}^{-1}\widetilde{\boldsymbol{g}}, \end{aligned} \tag{7}$$

which can be solved for δ_0^* if $\underline{\mathbf{H}}$ is negative definite. If Eq. (7) has multiple solutions, Venzon and Moolgavkar (1988) choose the one that minimizes $\boldsymbol{\delta}$ according to some norm. Our algorithm RVM applies a different procedure and chooses the root that minimizes the distance to $\theta_0^{\max}$ without stepping into a region in which the approximation (5) is inaccurate. In Sect. 2.5, we provide further details and discuss the case in which Eq. (7) has no real solutions.

After each iteration, $\boldsymbol{\theta}$ is updated according to the above results:

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} + \boldsymbol{\delta}^*. \tag{8}$$

If $\ell\left(\boldsymbol{\theta}^{(i+1)}\right) \approx \ell^*$ and $\frac{\partial \ell}{\partial \widetilde{\boldsymbol{\theta}}}\left(\boldsymbol{\theta}^{(i+1)}\right) \approx 0$ up to the desired precision, the search is terminated and $\boldsymbol{\theta}^{(i+1)}$ is returned.

The need to extend the original algorithm VM outlined above comes from the following issues: (1) The quadratic

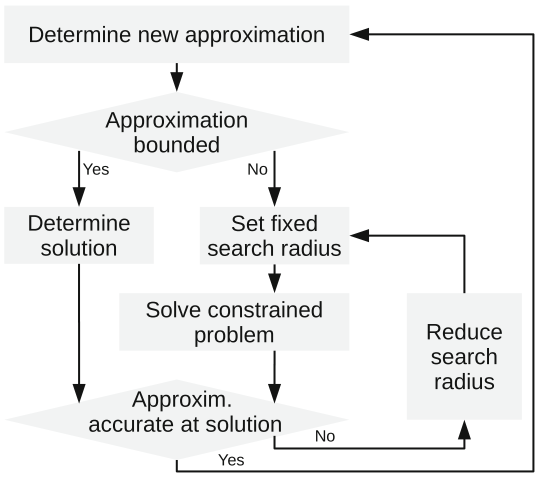

Fig. 1 Flow chart for RVM. The procedure is repeated until the termination criterion is met and the result is returned

approximation $\hat{\ell}$ may be imprecise far from the approximation point. In extreme cases, updating $\boldsymbol{\theta}$ as suggested could take us farther away from the target $\boldsymbol{\theta}^*$ rather than closer to it. (2) The approximation $\hat{\ell}$ may be constant in some directions or be unbounded above. In these cases, we may not be able to identify unique solutions for δ_0 and $\widetilde{\boldsymbol{\delta}}$, and the gradient criterion in condition 2 may not characterize a maximum but a saddle point or a minimum. (3) The limited precision of numerical operations can result in discontinuities corrupting the results of VM and hinder the algorithm from terminating.

To circumvent these problems, we introduce a number of extensions to VM. First, we address the limited precision of the Taylor approximation $\hat{\ell}$ with a trust region approach (Conn et al. 2000). That is, we constrain our search for $\boldsymbol{\delta}^*$ to a region in which the approximation $\hat{\ell}$ is sufficiently accurate. Second, we choose some parameters freely if $\hat{\ell}$ is constant in some directions and solve constrained maximization problems if $\hat{\ell}$ is not bounded above. In particular, we detect cases in which ℓ_{PL} approaches an asymptote above ℓ^*, which means that θ_0 is not estimable. Lastly, we introduce a method to identify and jump over discontinuities as appropriate. An overview of the algorithm is depicted as flow chart in Fig. 1. Below, we describe each of our extensions in detail.

2.2 The trust region

In practice, the quadratic approximation (5) may not be good enough to reach a point close to $\boldsymbol{\theta}^*$ within one step. In fact, since ℓ may be very "non-quadratic", we might obtain a parameter vector for which ℓ and $\frac{\partial \ell}{\partial \boldsymbol{\theta}}$ are farther from ℓ^* and $\mathbf{0}$ than in the previous iteration. Therefore, we accept changes in θ only if the approximation is sufficiently accurate in the new point.

In each iteration i, we compute the new parameter vector, compare the values of $\hat{\ell}$ and ℓ at the obtained point $\boldsymbol{\theta}^{(i)} + \boldsymbol{\delta}^*$, and accept the step if, and only if, $\hat{\ell}$ and ℓ are close together with respect to a given distance measure. If $\bar{\ell}$ is near the target ℓ^*, we may also check the precision of the gradient approximation $\frac{\partial \hat{\ell}}{\partial \boldsymbol{\theta}}$ to enforce timely convergence of the algorithm.

If we reject a step, we decrease the magnitude of the value $|\delta_0^*|$ obtained before, reduce the maximal admissible length r of the nuisance parameter vector and solve the constrained maximization problem

$$\widetilde{\boldsymbol{\delta}}^* = \underset{\widetilde{\boldsymbol{\delta}}:\,|\widetilde{\boldsymbol{\delta}}|\le r}{\operatorname{argmax}}\ \hat{\ell}^{\delta}\left(\delta_0, \widetilde{\boldsymbol{\delta}}\right). \tag{9}$$

As the quadratic subproblem (9) appears in classical trust-region algorithms, efficient solvers are available (Conn et al. 2000) and implemented in optimization software, such as in the Python package Scipy (Jones et al. 2001).

We check the accuracy of the approximation at the resulting point $\boldsymbol{\theta}^{(i)} + \boldsymbol{\delta}^*$, decrease the search radius if necessary, and continue with this procedure until the approximation is sufficiently precise. The metric and the tolerance applied to measure the approximation's precision may depend on how far the current log-likelihood $\bar{\ell}$ is from the target ℓ^*. We suggest suitable precision measures in Sect. 2.8.

Since it is typically computationally expensive to compute the Hessian $\underline{\mathbf{H}}$, we desire to take as large steps δ_0 as possible. However, it is also inefficient to adjust the search radius very often to find the maximal admissible δ_0^*. Therefore, RVM first attempts to make the unconstrained step given by Eqs. (6) and (7). If this step is rejected, RVM determines the search radius with a log-scale binary search between the radius of the unconstrained step and the search radius accepted in the previous iteration. If even the latter radius does not lead to a sufficiently precise result, we update δ_0^* and r by factors $\beta_0, \beta_1 \in (0, 1)$ so that $\delta_0^* \leftarrow \beta_0\delta_0^*$ and $r \leftarrow \beta_1 r$.

2.3 Linearly dependent parameters

The right hand side of Eq. (6) is defined only if the nuisance Hessian $\underline{\widetilde{\mathbf{H}}}$ is invertible. If $\underline{\widetilde{\mathbf{H}}}$ is singular, the maximum with respect to the nuisance parameters is not uniquely defined or does not exist at all. We will consider the second case in the next section and focus on the first case here.

If $\hat{\ell}$ has infinitely many maxima in the nuisance parameters, we can choose some nuisance parameters freely and consider a reduced system including the remaining independent parameters only. To that end, we check $\underline{\widetilde{\mathbf{H}}}$ for linear dependencies at the beginning of each iteration. We are interested in a minimal set S containing indices of rows and columns whose removal from $\underline{\widetilde{\mathbf{H}}}$ would make the matrix invertible. To compute S, we iteratively determine the ranks of sub-matrices of $\underline{\widetilde{\mathbf{H}}}$ using singular value decompositions (SVMs). SVMs are a well-known tool to identify the rank of a matrix and have also been applied to determine the number of identifiable parameters in a model (Eubank and Webster 1985; Viallefont et al. 1998).

We proceed as follows: first, we consider one row of $\underline{\widetilde{\mathbf{H}}}$ and determine its rank. Then, we continue by adding a second row, determine the rank of the new matrix and repeat the procedure until all rows, i.e. the full matrix $\underline{\widetilde{\mathbf{H}}}$, are considered. Whenever the matrix rank increases after addition of a row, this row is linearly independent from the previous rows. Conversely, the rows that do not increase the matrix rank are linearly dependent on other rows of $\underline{\widetilde{\mathbf{H}}}$. The indices of these rows form the set S. In general, the set of linearly dependent rows is not unique. Therefore, we consider the rows of $\underline{\widetilde{\mathbf{H}}}$ in descending order of the magnitudes of the corresponding gradient entries. This can help the algorithm to converge faster.

After S is determined, we need to check whether there is a parameter vector $\boldsymbol{\theta}^*$ satisfying requirements 1 and 2 from Sect. 2.1 for the approximation $\hat{\ell}$. Let $\underline{\widetilde{\mathbf{H}}}_{\mathrm{dd}}$ ("d" for "dependent") be the submatrix of $\underline{\mathbf{H}}$ that remains if all rows and columns corresponding to indices in S are removed from $\underline{\widetilde{\mathbf{H}}}$. Similarly, let $\underline{\widetilde{\mathbf{H}}}_{\mathrm{ff}}$ ("f" for "free") be the submatrix of $\underline{\widetilde{\mathbf{H}}}$ containing only the rows and columns corresponding to indices in S, and let $\underline{\widetilde{\mathbf{H}}}_{\mathrm{df}} = \underline{\widetilde{\mathbf{H}}}_{\mathrm{fd}}^{\top}$ be the matrix containing the rows whose indices are *not* in S and the columns whose indices *are* in S. Let us define $\widetilde{\boldsymbol{g}}_{\mathrm{d}}$, $\widetilde{\boldsymbol{g}}_{\mathrm{f}}$, $\widetilde{\boldsymbol{\delta}}_{\mathrm{d}}$, and $\widetilde{\boldsymbol{\delta}}_{\mathrm{f}}$ accordingly. If $\underline{\widetilde{\mathbf{H}}}_{\mathrm{dd}}$ is not negative definite, $\hat{\ell}$ is unbounded, and requirement 2 cannot be satisfied. Otherwise, we may attempt to solve

$$0 = \frac{\partial}{\partial \widetilde{\boldsymbol{\delta}}} \hat{\ell}^{\delta} \tag{10}$$

$$\iff$$

$$0 = \underline{\widetilde{\mathbf{H}}}_{\mathrm{dd}} \widetilde{\boldsymbol{\delta}}_{\mathrm{d}}^* + \underline{\widetilde{\mathbf{H}}}_{\mathrm{df}} \widetilde{\boldsymbol{\delta}}_{\mathrm{f}}^* + \widetilde{\mathbf{H}}_{0\mathrm{d}} \delta_0^* + \widetilde{\boldsymbol{g}}_{\mathrm{d}} \tag{11}$$

$$0 = \underline{\widetilde{\mathbf{H}}}_{\mathrm{df}}^{\top} \widetilde{\boldsymbol{\delta}}_{\mathrm{d}}^* + \underline{\widetilde{\mathbf{H}}}_{\mathrm{ff}} \widetilde{\boldsymbol{\delta}}_{\mathrm{f}}^* + \widetilde{\mathbf{H}}_{0\mathrm{f}} \delta_0^* + \widetilde{\boldsymbol{g}}_{\mathrm{f}}. \tag{12}$$

If equation system (11)–(12) has a solution, we can choose $\widetilde{\boldsymbol{\delta}}_{\mathrm{f}}^*$ freely. Setting $\widetilde{\boldsymbol{\delta}}_{\mathrm{f}}^* \leftarrow \mathbf{0}$ makes Eq. (11) equivalent to

$$\widetilde{\boldsymbol{\delta}}_{d}^* = -\underline{\widetilde{\mathbf{H}}}_{\mathrm{dd}}^{-1} \left(\widetilde{\mathbf{H}}_{0\mathrm{d}} \delta_0 + \widetilde{\boldsymbol{g}}_{\mathrm{d}} \right). \tag{13}$$

That is, we may set $\underline{\widetilde{\mathbf{H}}} \leftarrow \underline{\widetilde{\mathbf{H}}}_{\mathrm{dd}}$, $\widetilde{g} \leftarrow \widetilde{g}_{\mathrm{d}}$, $\widetilde{\boldsymbol{\delta}}^* \leftarrow \widetilde{\boldsymbol{\delta}}_{\mathrm{d}}^*$ for the remainder of the current iteration and proceed as usual, but leaving the free nuisance parameters unchanged: $\widetilde{\boldsymbol{\delta}}_{\mathrm{f}}^* = \mathbf{0}$. With the resulting δ_0^*, we check whether (12) holds approximately. If not, the log-likelihood is unbounded above. We consider this case in the next section.

An alternative way to proceed when $\underline{\widetilde{\mathbf{H}}}$ is singular is to apply a generalized matrix inverse in Eq. (6). For example, we could apply the Moore–Penrose inverse (Penrose 1955), which is well defined even for singular matrices. In tests, however, this approach appeared to be sensitive to a threshold parameter, and we obtained better results with the procedure described above. We present the alternative approach based on the Moore-Penrose inverse along with test results in Supplementary Appendix A.

2.4 Solving unbounded subproblems

In each iteration, we seek the nuisance parameters $\widetilde{\boldsymbol{\theta}}$ that maximize ℓ for the computed value of θ_0. The log-likelihood ℓ is bounded above if the MLE exists, which we presume. Nonetheless, the *approximate* log-likelihood $\hat{\ell}$ could be unbounded at times, which would imply that the approximation is imprecise for large steps. Since we cannot identify a global maximum of $\hat{\ell}$ if it is unbounded, we instead seek the point maximizing $\hat{\ell}$ in the range where $\hat{\ell}$ is sufficiently accurate.

If testing Eq. (12) in the previous section has not shown that $\hat{\ell}$ is unbounded above, we test the boundedness of $\hat{\ell}$ via a Cholesky decomposition on $-\underline{\widetilde{\mathbf{H}}}$. The decomposition succeeds if, and only if, $\underline{\widetilde{\mathbf{H}}}$ is negative definite, implying that $\hat{\ell}$ is bounded in the nuisance parameters. Otherwise, $\hat{\ell}$ is unbounded, since free parameters have been fixed and removed in the previous section, guaranteeing that $\underline{\widetilde{\mathbf{H}}}$ is nonsingular.

If $\hat{\ell}$ is unbounded, we set $\delta_0^* \leftarrow r_0$, $r \leftarrow r_1$ for some parameters $r_0, r_1 > 0$ and solve the maximization problem (9). The parameters r_0 and r_1 can be adjusted along with the trust region and saved for future iterations to efficiently identify the maximal admissible step. That is, we increase (or reduce) δ_0^* and r as long as (or until) $\hat{\ell}$ is sufficiently precise. In particular, we adjust the ratio of δ_0^* and r so that the likelihood increases: $\hat{\ell}^{\delta}\left(\delta_0^*, \widetilde{\boldsymbol{\delta}}^*\right) > \bar{\ell}$. At the end of the iteration, we set $r_0 \leftarrow \delta_0^*, r_1 \leftarrow r$ to start later iterations with appropriate step sizes.

2.5 Step choice for the parameter of interest

Whenever $\hat{\ell}$ has a unique maximum in the nuisance parameters, we compute δ_0^* by solving Eq. (7). This equation can have one, two, or no roots. To discuss how δ_0^* should be chosen in either of these cases, we introduce some helpful notation. First, we write $\hat{\ell}_{\mathrm{PL}}(\theta_0) := \max_{\tilde{\theta}} \hat{\ell}\left(\theta_0, \tilde{\theta}\right)$ for the profile log-likelihood function of the quadratic approximation. Furthermore, we write in accordance with previous notation

$$\hat{\ell}_{\mathrm{PL}}^{\delta}(\delta_0) := \hat{\ell}_{\mathrm{PL}}\left(\theta_0^{(i)} + \delta_0\right) = a\delta_0^2 + p\delta_0 + q + \ell^* \tag{14}$$

with $a := \frac{1}{2}\left(\mathrm{H}_{00} - \widetilde{\mathbf{H}}_0 \underline{\widetilde{\mathbf{H}}}^{-1} \widetilde{\mathbf{H}}_0\right)$, $p := g_0 - \widetilde{g}^{\top} \underline{\widetilde{\mathbf{H}}}^{-1} \widetilde{\mathbf{H}}_0$, and $q := \bar{\ell} - \frac{1}{2} \widetilde{g}^{\top} \underline{\widetilde{\mathbf{H}}}^{-1} \widetilde{g} - \ell^*$ [see Eq. (7)].

Our choices of δ_0^* attempt to increase θ_0 as much as possible while staying in a region in which the approximation

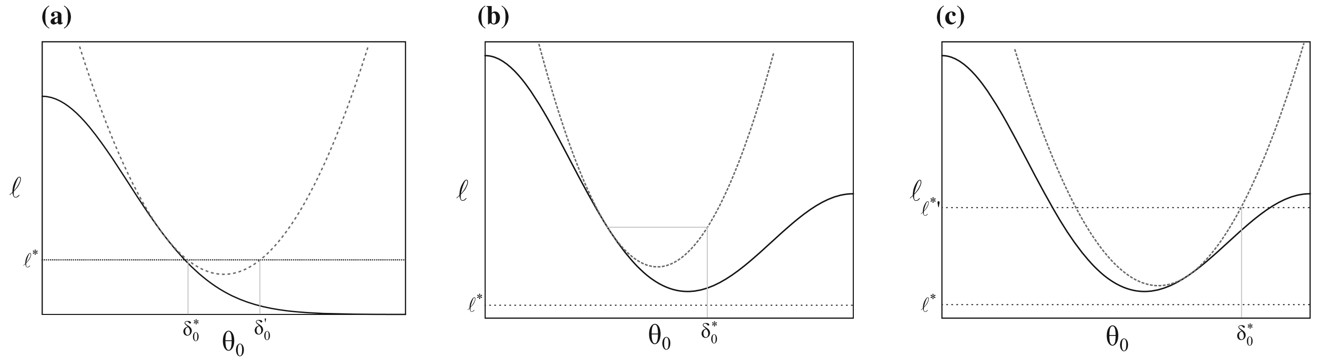

Fig. 2 Step choice for θ_0 in special cases. The figures depict the profile likelihood function ℓ_{PL} (solid black), quadratic approximation $\hat{\ell}_{\mathrm{PL}}$ (dashed parabola), and the threshold log-likelihood ℓ^*. **a** The approximation has two roots δ_0^* and δ_0'. Though the largest root of ℓ is searched, the smaller root of $\hat{\ell}$ is closest to the desired result. In fact, consistently choosing the larger root would let the algorithm diverge. **b** If ℓ_{PL} is decreasing but $\hat{\ell}_{\mathrm{PL}}$ does not assume the threshold value ℓ^*, we "jump" over the local minimum. **c** If ℓ_{PL} is increasing but $\hat{\ell}_{\mathrm{PL}}$ does not assume the threshold value ℓ^*, we reset the target value to an increased value $\ell^{*\prime}$

$\hat{\ell}$ is reasonably accurate. The specific step choice depends on the slope of the profile likelihood $\hat{\ell}_{\mathrm{PL}}^{\delta}$ and on whether we have already exceeded $\theta_0^{\max}$ according to our approximation, i.e. $\hat{\ell}_{\mathrm{PL}}^{\delta}(0) < \ell^*$. In Sects. 2.5.1–2.5.3 below, we assume that $\hat{\ell}_{\mathrm{PL}}^{\delta}(0) > \ell^*$. We discuss the opposite case in Sect. 2.5.4.

2.5.1 Case 1: decreasing profile likelihood

If the profile likelihood decreases at the approximation point, i.e. $p < 0$, we select the smallest positive root:

$$\delta_0^* = \begin{cases} -\frac{q}{p} & \text{if } a = 0 \\ -\frac{1}{2a}\left(p + \sqrt{p^2 - 4aq}\right) & \text{else.} \end{cases} \tag{15}$$

Choosing $\delta_0^* > 0$ ensures that the distance to the end point $\theta_0^{\max}$ decreases in this iteration. Choosing the smaller positive root increases our trust in the accuracy of the approximation and prevents potential convergence issues (see Fig. 2a).

If $\hat{\ell}_{\mathrm{PL}}^{\delta}$ has a local minimum above the threshold ℓ^*, Eq. (14) does not have a solution, and we may attempt to decrease the distance between $\hat{\ell}_{\mathrm{PL}}^{\delta}$ and ℓ^* instead. This procedure, however, may let RVM converge to a local minimum in $\hat{\ell}_{\mathrm{PL}}^{\delta}$ rather than to a point with $\hat{\ell}_{\mathrm{PL}}^{\delta} = \ell^*$. Therefore, we "jump" over the extreme point by doubling the value of δ_0^*. That is, we choose

$$\delta_0^* = -\frac{p}{a} \tag{16}$$

if $p^2 < 4aq$ (see Fig. 2b). This choice of δ_0^* ensures that we quickly return to a range where the profile likelihood function decreases while at the same time accounting for the scale of the problem by considering the curvature of the profile likelihood.

2.5.2 Case 2: increasing profile likelihood

If the profile likelihood increases at the approximation point, i.e. $p > 0$, Eq. (14) has a positive root if, and only if, $\hat{\ell}_{\mathrm{PL}}$ is concave down; $a < 0$. We choose this root whenever it exists:

$$\delta_0^* = -\frac{1}{2a}\left(p + \sqrt{p^2 - 4aq}\right). \tag{17}$$

However, if $\hat{\ell}_{\mathrm{PL}}$ grows unboundedly, Eq. (14) does not have a positive root. In this case, we change the threshold value ℓ^* temporarily to a value $\ell^{*\prime}$ chosen so that Eq. (14) has a solution with the updated threshold (see Fig. 2c). For example, we may set

$$\ell^{*\prime} := \max\left\{\hat{\ell}_{\mathrm{PL}}^{\delta}(0) + 1, \frac{\bar{\ell} + \ell\left(\hat{\boldsymbol{\theta}}\right)}{2}\right\}. \tag{18}$$

The first term in the maximum expression in (18) ensures that a solution exists; setting the threshold 1 unit higher than the current approximate value of the profile log-likelihood seems reasonable given that we typically consider the log-likelihood surface in a range where it is $\mathcal{O}(1)$ units below its maximum. The second term in the maximum expression permits us to take larger steps if we are far below the likelihood maximum. That way, we may reach local likelihood maxima faster. After resetting the threshold, we proceed as usual.

To memorize that we have changed the threshold value ℓ^*, we set a flag `maximizing` $\leftarrow$ `True`. In future iterations $j > i$, we set the threshold ℓ^* back to its initial value and `maximizing` $\leftarrow$ `False` as soon as $\ell\left(\boldsymbol{\theta}^{(j)}\right)$ falls below the initial threshold or $\hat{\ell}_{\mathrm{PL}}$ is concave down at the approximation point $\boldsymbol{\theta}^{(j)}$.

2.5.3 Case 3: constant profile likelihood

If the profile likelihood has a local extremum at the approximation point, i.e. $p = 0$, $a \neq 0$, we proceed as in cases 1 and 2: if $a > 0$, we proceed as if $\hat{\ell}_{\mathrm{PL}}$ were increasing, and if $a < 0$, we proceed as if $\hat{\ell}_{\mathrm{PL}}$ were decreasing. However, the approximate profile likelihood could also be constant, $a = p = 0$. In this case, we attempt to make a very large step to check whether we can push θ_0 arbitrarily far. In Sect. 2.6, we discuss this procedure in greater detail.

2.5.4 Profile likelihood below the threshold

If the profile likelihood at the approximation point is below the threshold, $\hat{\ell}^{\delta}_{\mathrm{PL}}(0) < \ell^*$, we always choose the smallest possible step:

$$\delta_0^* = \begin{cases} -\frac{1}{2a}\left(p + \sqrt{p^2 - 4aq}\right) & \text{if } a \neq 0,\ p < 0 \\ -\frac{q}{p} & \text{if } a = 0,\ p \neq 0 \\ -\frac{1}{2a}\left(p - \sqrt{p^2 - 4aq}\right) & \text{if } a \neq 0,\ p > 0. \end{cases} \tag{19}$$

This shall bring us to the admissible parameter region as quickly as possible.

As RVM rarely steps far beyond the admissible region in practice, Eq. (19) usually suffices to define δ_0^*. Nonetheless, if we find that $\hat{\ell}^{\delta}_{PL}$ has a local maximum below the threshold, i.e. $p^2 < 4qa$, we may instead maximize $\hat{\ell}^{\delta}_{PL}$ as far as possible:

$$\delta_0^* = -\frac{p}{2a}. \tag{20}$$

If we have already reached a local maximum ($p \approx 0$), we cannot make a sensible choice for δ_0. In this case, we may recall the iteration $k := \underset{j:\, \ell(\theta^{(j)}) \geq \ell^*}{\operatorname{argmax}} \theta_0^{(j)}$, in which the largest admissible θ_0 value with $\ell\left(\theta^{(k)}\right) \geq \ell^*$ has been found so far, and conduct a binary search between $\theta^{(i)}$ and $\theta^{(k)}$ until we find a point $\theta^{(i+1)}$ with $\ell\left(\theta^{(i+1)}\right) \geq \ell^*$.

2.6 Identifying inestimable parameters

If the considered parameter is not estimable and the profile log-likelihood ℓ_{PL} never falls below the threshold ℓ^*, RVM may not converge. However, often it is possible to identify inestimable parameters by introducing a step size limit $\delta_0^{\max}$. If the computed step exceeds the maximal step size, $\delta_0^* > \delta_0^{\max}$ and the current function value exceeds the threshold value, i.e. $\bar{\ell} \geq \ell^*$, we set $\delta_0^* := \delta_0^{\max}$ and compute the corresponding nuisance parameters. If the resulting log-likelihood $\ell\left(\theta^{(i)} + \boldsymbol{\delta}^*\right)$ is not below the threshold ℓ^*, we let the algorithm terminate, raising a warning that the parameter θ_0 is not estimable. If $\ell\left(\theta^{(i)} + \boldsymbol{\delta}^*\right) < \ell^*$, however, we cannot draw this conclusion and decrease the step size until the approximation is sufficiently close to the original function.

The criterion suggested above may not always suffice to identify inestimable parameters. For example, if the profile likelihood is constant but the nuisance parameters maximizing the likelihood change non-linearly, RVM may not halt. For this reason, and also to prevent unexpected convergence issues, it is advisable to introduce an iteration limit to the algorithm. If the iteration limit is exceeded, potential estimability issues may be investigated further.

2.7 Discontinuities

RVM is based on quadratic approximations and requires therefore that ℓ is differentiable twice. Nonetheless, discontinuities can occur due to numerical imprecision even if the likelihood function is continuous in theory. Though we may still be able to compute the gradient g and the Hessian $\underline{\mathbf{H}}$ in these cases, the resulting quadratic approximation will be inaccurate even if we take very small steps. Therefore, these discontinuities could hinder the algorithm from terminating.

To identify discontinuities, we define a minimal step size ϵ_{step}, which may depend on the gradient $\boldsymbol{g}$. If we reject a step with small length $\left|\boldsymbol{\delta}^*\right| \leq \epsilon_{\text{step}}$, we may conclude that ℓ is discontinuous at the current approximation point $\boldsymbol{\theta}^{(i)}$. To determine the set D of parameters responsible for the issue, we decompose $\boldsymbol{\delta}^*$ into its components. We initialize $D \leftarrow \emptyset$ and consider, with the jth unit vector $\boldsymbol{e}_j$, the step $\boldsymbol{\delta}^{*\prime} := \sum_{j \leq k,\, j \neq D} \boldsymbol{e}_j \delta_j^*$ until $\hat{\ell}^{\delta}\left(\boldsymbol{\delta}^{*\prime}\right) \not\approx \ell^{\delta}\left(\boldsymbol{\delta}^{*\prime}\right)$ for some $k < n$. When we identify such a component, we add it to the set D and continue the procedure.

If we find that ℓ is discontinuous in θ_0, we check whether the current nuisance parameters maximize the likelihood, i.e. ℓ is bounded above and $\widetilde{\boldsymbol{g}}$ is approximately $\mathbf{0}$. If the nuisance parameters are not optimal, we hold θ_0 constant and maximize ℓ with respect to the nuisance parameters. Otherwise, we conclude that the profile likelihood function has a jump discontinuity. In this case, our action depends on the current log-likelihood value $\bar{\ell}$, the value of ℓ at the other end of the discontinuity, and the threshold ℓ^*.

- If $\ell\left(\boldsymbol{\theta}^{(i)} + \boldsymbol{e}_0\delta_0^*\right) \geq \ell^*$ or $\ell\left(\boldsymbol{\theta}^{(i)}\right) < \ell\left(\boldsymbol{\theta}^{(i)} + \boldsymbol{e}_0\delta_0^*\right)$, we accept the step regardless of the undesirably large error.
- If $\ell\left(\boldsymbol{\theta}^{(i)} + \boldsymbol{e}_0\delta_0^*\right) < \ell^*$ and $\ell\left(\boldsymbol{\theta}^{(i)}\right) \geq \ell^*$, we terminate and return $\theta_0^{(i)}$ as the bound of the confidence interval.
- Otherwise, we cannot make a sensible step and try to get back into the admissible region by conducting the binary search procedure we have described in Sect. 2.5.4.

If ℓ is discontinuous in variables other than θ_0, we hold the variables constant whose change decreases the likelihood and repeat the iteration with a reduced system. After a given number of iterations, we release these parameters again, as θ may have left the point of discontinuity.

Since we may require that not only $\hat{\ell}$ but also its gradient are well approximated, a robust implementation of RVM should also handle potential gradient discontinuities. The nuisance parameters causing the issues can be identified analogously to the procedure outlined above. All components in which the gradient changes its sign from positive to negative should be held constant, as the likelihood appears to be in a local maximum in these components. The step in the remaining components may be accepted regardless of the large error.

2.8 Suitable parameters and distance measures

The efficiency of RVM depends on the distance measures and parameters applied when assessing the accuracy of the approximation and updating the search radius of the constrained optimization problems (9). If the precision measures are overly conservative, then many steps will be needed to find $\boldsymbol{\theta}^*$. If the precision measure is too liberal, in turn, RVM may take detrimental steps and might not even converge.

We suggest the following procedure: (1) we always accept forward steps with $\delta_0^* \geq 0$ if the true likelihood is larger than the approximate likelihood: $\ell^\delta(\boldsymbol{\delta}^*) \geq \hat{\ell}^\delta(\boldsymbol{\delta}^*)$. (2) If the approximate likelihood function is unbounded, we require that the likelihood increases: $\ell^\delta(\boldsymbol{\delta}^*) \geq \bar{\ell}$. This requirement helps RVM to return quickly to a region in which the approximation is bounded. However, if the step size falls below the threshold used to detect discontinuities, we may accept satisfactory precise steps even when the likelihood does not increase. This prevents the algorithm from seeking potential discontinuities even though the approximation is precise. (3) If we are outside the admissible region, i.e. $\bar{\ell} < \ell^*$, we enforce that we get closer to the target likelihood: $\left|\ell^\delta(\boldsymbol{\delta}^*) - \ell^*\right| < \left|\bar{\ell} - \ell^*\right|$. This reduces potential convergence issues. (4) We require that

$$\frac{\left|\hat{\ell}^\delta(\boldsymbol{\delta}^*) - \ell^\delta(\boldsymbol{\delta}^*)\right|}{\left|\bar{\ell} - \ell^*\right|} \leq \gamma \tag{21}$$

for a constant γ. That is, the required precision depends on how close we are to the target. This facilitates fast convergence of the algorithm. The constant $\gamma \in (0, 1)$ controls how strict the precision requirement is. In tests, $\gamma = \frac{1}{2}$ appeared to be a good choice. (5) If we are close to the target, $\ell^\delta(\boldsymbol{\delta}^*) \approx \ell^*$, we also require that the gradient estimate is precise:

$$\frac{\left|\frac{\partial \hat{\ell}^\delta}{\partial \tilde{\theta}}(\boldsymbol{\delta}^*) - \frac{\partial \ell^\delta}{\partial \tilde{\theta}}(\boldsymbol{\delta}^*)\right|}{|\boldsymbol{g}|} \leq \gamma. \tag{22}$$

This constraint helps us to get closer to a maximum in the nuisance parameters. Here, we use the $\mathcal{L}_2$ norm.

When we reject a step because the approximation is not sufficiently accurate, we adjust δ_0^* and solve the constrained maximization problem (9) requiring $\left|\widetilde{\boldsymbol{\delta}}\right| \leq r$. To ensure that the resulting step does not push the log-likelihood below the target ℓ^*, the radius r should not be decreased more strongly than δ_0^*. In tests, adjusting r by a factor $\beta_1 := \frac{2}{3}$ whenever δ_0^* is adjusted by factor $\beta_0 := \frac{1}{2}$ yielded good results. Aside from accepting or rejecting proposed steps, the algorithm evaluates the accuracy of various equations with some tolerance. We suggest to introduce a single tuning parameter ϵ_{tol} to control the accuracy requirement. In tests, we found that a generous bound of $\epsilon_{\text{tol}} = 0.001$ makes the algorithm robust against errors arising, for example, if almost singular matrices are inverted.

The optimal value for the minimal step size ϵ_{step}, used to detect discontinuities due to numerical errors, depends on the expected accuracy of the gradient and Hessian of the log-likelihood function. We suggest a default value of 10^{-5}. The maximal step size $\delta_0^{\max}$, used to classify parameters as inestimable, should be chosen as large as possible without leading to numerical issues. That way, the criterion becomes less dependent on the scale of the parameter. We suggest a value of $\delta_0^{\max} = 10^{10}$, as it is far beyond the typical range of parameters in practical problems. Nonetheless, the parameter needs to be adjusted if very large parameter values may be expected.

The distance measures and parameters given above are meant to be applicable in a wide range of problems without further fine tuning. We list the tuning parameters along with the suggested values in Table 1.

2.9 Confidence intervals for functions of parameters

Often, modelers are interested in confidence intervals for functions $f(\boldsymbol{\theta})$ of the parameters. A limitation of VM and RVM is that such confidence intervals cannot be computed directly with these algorithms. However, this problem can be solved approximately by considering a slightly changed likelihood function. We aim to find

$$\phi^{\max} = \max_{\boldsymbol{\theta} \in \Theta:\, \ell(\boldsymbol{\theta}) \geq \ell^*} f(\boldsymbol{\theta}) \tag{23}$$

Table 1 Tuning parameters along with suggested values that typically yield good results in practice

Parameter	Explanation	Suggested value
γ	Accepted error of the approximate log-likelihood relative to the remaining distance to the target	$\frac{1}{2}$
ϵ_{tol}	Tolerance for equations	10^{-3}
ϵ_{step}	Minimal step size; used to detect discontinuities due to numerical errors	10^{-5}
β_0	Update factor for the step in the parameter of interest if a step is rejected	$\frac{1}{2}$
β_1	Update factor for the trust region radius if a step is rejected	$\frac{2}{3}$
$\delta_0^{\max}$	Maximal step size $\delta_0^{\max}$; used to classify parameters as inestimable	10^{10}

or the respective minimum. Define

$$\check{\ell}(\phi, \boldsymbol{\theta}) := \ell(\boldsymbol{\theta}) - \frac{1}{2}\left(\frac{f(\boldsymbol{\theta}) - \phi}{\varepsilon}\right)^2 \chi^2_{1,1-\alpha}, \tag{24}$$

with a small constant ε. Consider the altered maximization problem

$$\check{\phi}^{\max} = \max_{\boldsymbol{\theta}\in\Theta:\, \check{\ell}(\phi,\boldsymbol{\theta})\geq \ell^*} \phi, \tag{25}$$

which can be solved with VM or RVM.

We argue that a solution to (25) is an approximate solution to (23), with an error bounded by ε. Let $\left(\phi^{\max}, \boldsymbol{\theta}^*\right)$ be a solution to problem (23) and $\left(\check{\phi}^{\max}, \check{\boldsymbol{\theta}}^*\right)$ a solution to problem (25). Since $\phi^{\max} = f\left(\boldsymbol{\theta}^*\right)$, it is $\check{\ell}\left(\phi^{\max}, \boldsymbol{\theta}^*\right) = \ell\left(\boldsymbol{\theta}^*\right) \geq \ell^*$. Therefore, $\left(\phi^{\max}, \boldsymbol{\theta}^*\right)$ is also a feasible solution to (25), and it follows that $\check{\phi}^{\max} \geq \phi^{\max}$. At the same time, $\check{\ell}(\phi, \boldsymbol{\theta}) \leq \ell(\boldsymbol{\theta})$, which implies that $f\left(\check{\boldsymbol{\theta}}^*\right) \leq f\left(\boldsymbol{\theta}^*\right)$, since $\boldsymbol{\theta}^*$ maximizes f over a domain larger than the feasibility domain of (25). In conclusion, $f\left(\check{\boldsymbol{\theta}}^*\right) \leq f\left(\boldsymbol{\theta}^*\right) = \phi^{\max} \leq \check{\phi}^{\max}$. Lastly,

$$\begin{aligned}\ell^* = \ell\left(\hat{\boldsymbol{\theta}}\right) - \frac{1}{2}\chi^2_{1,1-\alpha} &\leq \check{\ell}\left(\check{\phi}^{\max}, \check{\boldsymbol{\theta}}^*\right)\\ &= \ell\left(\check{\boldsymbol{\theta}}^*\right) - \frac{1}{2}\left(\frac{f\left(\check{\boldsymbol{\theta}}^*\right) - \check{\phi}^{\max}}{\varepsilon}\right)^2 \chi^2_{1,1-\alpha}.\end{aligned} \tag{26}$$

Simplifying (26) yields $\left|f\left(\check{\boldsymbol{\theta}}^*\right) - \check{\phi}^{\max}\right| \leq \varepsilon$. Thus, $\left|\phi^{\max} - \check{\phi}^{\max}\right| \leq \varepsilon$.

Though it is possible to bound the error by an arbitrarily small constant ε in theory, care must be taken if the function $f(\boldsymbol{\theta})$ is not well-behaved, i.e. strongly nonlinear. In these cases, overly small values for ε may slow down convergence. Clearly, considering an objective function with an added parameter increases the computational effort to compute the Hessian matrix required for RVM. Note, however, that the Hessian of the altered likelihood $\check{\ell}$ can be computed easily if the Hessian of the original likelihood $\check{\ell}$ and the derivatives of the function f are known. Therefore, determining confidence intervals for functions of parameters via solving the altered problem (4) may not be computationally harder than determining confidence intervals for parameters.

Observe that the suggested procedure may seem to resemble the approach of Neale and Miller (1997), who also account for constraints by adding the squared error to the target function. However, unlike Neale and Miller (1997), the approach suggested above bounds the error in the confidence interval bound, not the error of the constraint. Furthermore, we do not square the log-likelihood function, which would worsen nonlinearities and could thus make optimization difficult. Therefore, our approach is less error-prone than the method by Neale and Miller (1997).

3 Tests

To compare the presented algorithm to existing methods, we applied RVM (with the suggested tuning parameters), the classic VM, and five other algorithms to benchmark problems and compared the robustness and performance of the approaches. Below we review the implemented methods. Then we introduce the benchmark problems, before we finally present the benchmark results.

3.1 Methods implemented for comparison

Besides RVM and VM, we implemented three methods that repeatedly evaluate the profile likelihood function (grid search, quadratic bisection, and binary search) and two methods that search for the confidence intervals directly (sequential least squares programming and the method by Neale and Miller). We implemented all methods in the programming language Python version 3.7 and made use of different optimization routines implemented or wrapped in the scientific computing library Scipy (Jones et al. 2001).

First, we implemented a grid search for the confidence bounds. The approach uses repeated Lagrangian constrained optimizations and may resemble the method by DiCiccio and Tibshirani (1991); however, rather than implementing the algorithm by DiCiccio and Tibshirani (1991), we applied the constrained optimization algorithm by Lalee et al. (1998), which is a trust-region approach and may thus be more robust than the method by DiCiccio and Tibshirani (1991). Furthermore, the algorithm by Lalee et al. (1998) was readily implemented in Scipy.

We conducted the grid search with a naive step size of 0.2, which we repeatedly reduced by factor 2 close to the threshold log-likelihood ℓ^* until the desired precision was achieved. In practical applications, modelers may have prior knowledge of the scale of the parameter and could choose the step size accordingly. To test the general applicability of the algorithm, however, we assumed that no prior knowledge is available and held the step size constant. To account for unidentifiable parameters, we attempted one large step (1000 units) if the algorithm did not terminate in the given iteration limit. We considered a parameter as unidentifiable if this step yielded a log-likelihood above the target value ℓ^*.

Second, we implemented a quadratic bisection method for root finding on ℓ_{PL} (cf. Ren and Xia 2019). Initially we chose a step size of 1. Afterwards, we computed the step of θ_0 based on a quadratic interpolation between the MLE $\hat{\theta}_0$, the maximal value of θ_0 for which we found $\ell_{\mathrm{PL}}(\theta_0) > \ell^*$ and the smallest identified value of θ_0 with $\ell_{\mathrm{PL}}(\theta_0) < \ell^*$. Until a point θ_0 with $\ell_{\mathrm{PL}}(\theta_0) < \ell^*$ was identified, we interpolated ℓ_{PL} between $\hat{\theta}_0$ and the two largest evaluated values θ_0. When only two points were available or the approximation of ℓ_{PL} did not assume the target value, we introduced the additional constraint $\frac{d^2\ell_{\mathrm{PL}}}{d\theta_0^2} = 0$. Using a quadratic rather than a linear interpolation for bisection has the advantage that the algorithm converges faster if the profile log-likelihood function is convex or quadratic. To evaluate ℓ_{PL}, we applied sequential least squares programming (Kraft 1988), which is the default method for constrained optimization in Scipy.

Third, we implemented a binary search with an initial step of 1. Until a value θ_0 with $\ell_{\mathrm{PL}}(\theta_0) < \ell^*$ was found, we increased the step δ_0 by factor 10. This preserves the logarithmic runtime of the algorithm if the problem has a solution. To broaden the range of tested internal optimization routines, we used a different method to evaluate ℓ_{PL} than in the bisection method: we fixed θ_0 at the desired value and performed an unconstrained optimization on the nuisance parameters. Here, we used the quasi-Newton method by Broyden, Fletcher, Goldfarb, and Shanno (BFGS; see Nocedal et al. 2006, p. 136).

To test methods that search for the confidence interval end points directly, we solved problem (4) with sequential least squares programming (Kraft 1988). Furthermore, we implemented the approximate method by Neale and Miller (1997). They transform the constrained maximization problem (9) to an unconstrained problem by considering the sum of the parameter of interest θ_0 and the squared error between the target ℓ^* and the log-likelihood. Minimization of this target function yields a point in which the target log-likelihood is reached approximately and the parameter of interest is minimal. Again, we used the method BFGS for minimization (see above).

Finally, we implemented Wald's method to assess the need to apply any profile likelihood method.

3.2 Test problem with empirical data set

To test the applicability of RVM in real-world problems, we considered a model for in-cell dynamics of the protein histone H1.2. The model, a dynamical system with 6 free parameters, was developed by Contreras et al. (2018), who fitted the model to laboratory data (Raghuram et al. 2010) via least squares, supposing normally distributed errors. The data set consists of 59 observations. Contreras et al. (2018) constrained the parameters to the positive range, as negative parameter values were not biologically reasonable. In the admissible parameter range, the likelihood function exhibits four local maxima with equal likelihood, each corresponding to a parameter vector with two zero entries. As a result, Wald's method does not yield useful confidence intervals at either of the likelihood maxima, and Contreras et al. (2018) report parameter estimates for simplified models only, in which zero-parameters were fixed. Though considering reduced models is an appropriate response to zero estimates, it remains unclear which of the four model candidates should be chosen (Contreras et al. 2018). In light of this uncertainty, it is of interest to compute confidence intervals for the parameters without choosing any of the four model candidates a priori.

We applied all described algorithms to the histone H1.2 model and computed confidence intervals for the parameters. To constrain the parameters to the positive range, we considered the log-transformed parameter space. We initialized the algorithms close to one of the MLEs reported by Contreras et al. (2018), replacing zeros with small positive values (e^{-10}). For the algorithms that repeatedly evaluate the profile likelihood function (grid search, quadratic bisection, and binary search), we set the desired result accuracy to 10^{-5}. Furthermore, we let these algorithms terminate if they found a lower confidence interval bound below -1000 in the log-space. Whenever necessary, we computed gradients and Hessian matrices with the central difference method implemented in the Python package numdifftools (Brodtkorb and D'Errico 2019). We collected the computed confidence intervals and recorded how many likelihood evaluations the algorithms needed.

3.3 Benchmark problems with simulated data

Though the performance of the algorithms in a specific use case is a valuable benchmark, a single example may not suffice to draw general conclusions. To test the implemented methods under a broader range of scenarios and investigate how model characteristics affect the algorithms' performances, we applied them to a class of benchmark problems with variable parameter number and data set size. We considered a logistic regression problem with n count data covariates c_{ij}, $j \in \{1, \ldots, n\}$ for each data point $i \in \{1, \ldots, N\}$. We assumed that the impact of each covariate levels off at high values and considered therefore the transformed covariates $c_{ij}^{\alpha_j}$ with $\alpha \in (0, 1)$. This is not only reasonable in many real world problems but also makes likelihood maximization a computationally challenging problem if not enough data are available to make the model locally approximately normal. Hence, this scenario gives insights into the performance of the implemented methods in challenging realistic problems. The benchmark model's probability mass function for a data point X_i was thus given by

$$\mathbb{P}(X_i = 1) = \left(1 + \exp\left(-\beta_0 - \sum_j \beta_j c_{ij}^{\alpha_j}\right)\right)^{-1} \tag{27}$$

and $\mathbb{P}(X_i = 0) = 1 - \mathbb{P}(X_i = 1)$.

We drew the covariate values randomly from a negative binomial distribution with mean 5 and variance 10. The negative binomial distribution is commonly used to model count data (Gardner et al. 1995) and thus suited to represent count covariates. To simulate the common case that covariates are correlated, we furthermore drew the value for every second covariate from a binomial distribution with the respective preceding covariate as count parameter. That is, for uneven j,

$$c_{i,j+1} \sim \text{Binomial}\big(c_{i,j}, p\big), \tag{28}$$

with $p = 0.2$ in our simulations. To avoid numerical problems arising when covariates with value 0 are raised to the power 0, we added a small positive perturbation to the count values. That way, we achieved that 0^0 was defined to be 1. We chose the parameters α_j and β_j so that the data were balanced, i.e. the frequency of 0s and 1s was approximately even. Refer to Supplementary Appendix B for the parameter values we used.

3.4 Benchmark procedure

We considered three classes of the benchmark problem described above: models with 1 covariate (3 parameters), models with 5 covariates (11 parameters), and generalized linear models (GLM) with 10 covariates, in which the powers α_j were set to 1 (11 parameters). Furthermore, we varied the sizes of the simulated data sets, ranging between $N = 500$ and $N = 10{,}000$ for the models with transformed covariates and $N = 50$ and $N = 1000$ for the GLM. In Fig. 3, we depict the impact of N on the shape of the likelihood function and thus the difficulty of the problem.

For each considered set of parameters, we generated 200 realizations of covariates and training data from the model described in the previous section. We determined the maximum likelihood estimator by maximizing the log-likelihood with the method BFGS and refined the estimate with an exact trust region optimizer (Conn et al. 2000). Then, we applied each of the implemented algorithms to find confidence intervals for all parameters for each data set and determined the algorithms' success rates and efficiencies.

As the likelihood functions of the tested models decrease drastically at $\alpha_j = 0$, potentially causing some algorithms to fail, we constrained the α_j to non-negative values. To that end, we considered transformed parameters $\alpha'_j := \ln\big(\exp\big(\alpha_j\big) - 1\big)$. Nonetheless, we evaluated the results of the tested algorithms based on the back-transformed parameters α_j.

We measured the algorithms' success based on their ability to solve problem (4) rather than their capability to determine the true confidence intervals for the parameters. Though profile likelihood confidence intervals are usually highly accurate, they rely on the limiting distribution of the likelihood ratio statistic. Therefore, algorithms could fail to solve optimization problem (4) but, by coincidence, return a result close to the true confidence interval bound and vice versa. To exclude such effects and circumvent the high computational effort required to determine highly precise confidence intervals with sampling methods, we determined the "true" confidence interval bound by choosing the widest confidence interval bound obtained by either of the tested methods provided it was admissible, i.e. $\ell(\theta^{\max}) \geq \ell^*$ up to a permissible error of 0.001.

We considered an algorithm successful if (1) the returned result was within a ±5% range of the true confidence interval bound or had an error below 0.001, and (2) the algorithm reported convergence. That is, to be deemed successful, an algorithm had to both return the correct result and also claim that it found the correct solution. The latter constraint ensures that if none of the algorithms converges successfully, even the one with the best result is not considered successful.

As many of the tested methods rely on general optimizers without specific routines to identify situations with divergent solutions, we considered parameters with confidence interval bounds exceeding $[-1000, 1000]$ in the transformed parameter space as unbounded. Consequently, all algorithms returning a larger confidence interval were considered successful.

We limited the runtime of all methods except the pre-implemented optimizers by introducing a step limit of 200. If convergence was not reached within this number of steps, the algorithms were viewed unsuccessful except for the case with inestimable parameters.

To test whether some methods tend to return misleading results, we determined the mean absolute error between the returned and the true confidence interval bounds when algorithms reported success. As this quantity can be dominated by outliers, we also determined the mean of all errors below 10 and the frequency of errors beyond 10, below called "large errors".

We measured the computational speed of the different methods by recording the number of function evaluations required until termination. This provides us with precise benchmark results independent of hardware and implementation details. To display a potential trade-off between robustness (success rate) and speed (number of function evaluations), we did not consider cases in which convergence was not reached. That way, internal stopping criteria did not affect the results.

The specific advantage of some optimization algorithms is in not requiring knowledge of the Hessian matrix. As computing the Hessian is necessary for RVM and may reduce the algorithm's performance compared to other methods, we included the number of function evaluations required to determine the Hessian and the gradient in the recorded count of function evaluations. We computed gradients and Hessian matrices with a complex step method (Lai et al. 2005) implemented in the Python package numdifftools (Brodtkorb and D'Errico 2019).

3.5 Results

To provide an impression of how RVM acts in practice, we plotted the trajectory of RVM along with ancillary function evaluations for one of the benchmark models in Fig. 3. It is visible that the algorithm stays on the "ridge" of the likelihood surface even if the admissible region is strongly curved.

A summary of the results for the histone H1.2 model is given in Table 2. The algorithms RVM, grid search, and bisection successfully identified all confidence interval bounds up to the desired precision. Out of these algorithms, the bisection method was the most efficient with less than 80,000 required likelihood evaluations, followed by RVM with almost three times as many likelihood evaluations. The binary search and the method by Neale and Miller failed to identify 1 and 2 interval bounds, respectively. The latter, requiring little more than 70,000 likelihood evaluations, was the most efficient method with more than 10 out of 12 correctly identified confidence interval bounds. The constrained maximization algorithm identified only 7 bounds correctly; VM and Wald's method had very low success with 1 and 0 correctly identified bounds, respectively. The specific confidence interval bounds computed via the methods can be found in Supplementary Appendix C.

Table 2 Success and efficiency of the tested methods for the histone H1.2 model

Method	Successes	Likelihood evaluations
RVM	12 (100%)	214,818
VM	1 (8%)	199,556
Grid search	12 (100%)	1,527,067
Bisection	12 (100%)	75,696
Binary search	11 (92%)	399,680
Neale–Miller	10 (83%)	73,727
Constr. max.	7 (58%)	71,993
Wald	0 (0%)	0

"Successes" refers to the number of confidence interval bounds for which the algorithms returned results within a ±5% tolerance around the correct values, respectively. Note that Wald's method does not require any likelihood evaluations if the observed information matrix is known

For the benchmark problems, the best results were often obtained via RVM (see Fig. 4). In all considered scenarios, RVM was the algorithm with the highest success rate, which never fell below 90% (second best: binary search, 52%). In scenarios with small data sets, the success rate of RVM was up to 37 percent points higher than that of any other method. At the same time, RVM was among the fastest algorithms. In scenarios with large data sets, RVM often converged within three iterations. Furthermore, RVM was quick in the 3 parameter model, in which the Hessian matrix is easy to compute. In the scenario with transformed covariates and 11 parameters, RVM required about three times as many likelihood evaluations as the fastest algorithm but had a more than 56% higher success rate. The error in the results returned by RVM was consistently low compared to other methods. The proportion of large errors was always below 1%, and the mean error excluding these outliers never exceeded 0.05.

The algorithms that require repeated evaluations of the profile likelihood function performed second best in terms of the success rate. Except for the GLM with 50 data points, the binary search, the grid search, and the bisection method consistently had success rates above 70%, and the success rate increased with the size of the considered data set. However, these algorithms also required more function evaluations than other methods. In fact, the grid search was more than 5 times slower than any other algorithm. The binary search was slightly less efficient than the bisection method, which exploits the approximately quadratic shape of the profile likelihood function if many data are available. In scenarios with large data sets, the bisection method was among

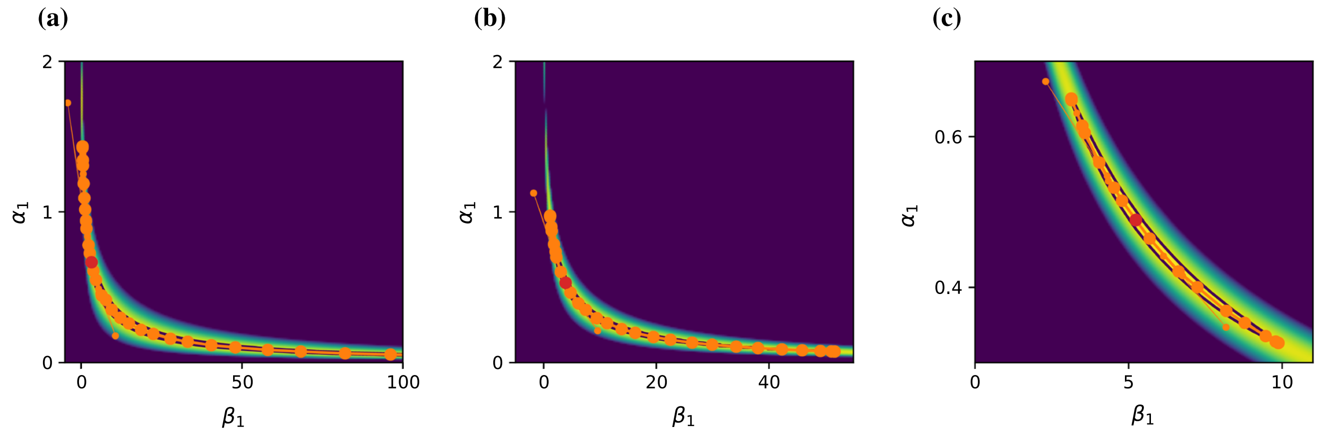

Fig. 3 Likelihood surface of the 3-parameter benchmark model with different data set sizes N. As N increases, the confidence region becomes smaller and closer to an elliptic shape. The orange dots depict the accepted (large dots) and rejected (small dots) steps of RVM searching for a confidence interval for β_1. RVM follows the ridge of the likelihood surface. The red dot shows the location of the MLE $\hat{\theta}$. The background color depicts the respective maximal log-likelihood for the given α_1 and β_1 ranging from $\leq \hat{\ell} - 50$ (dark blue) to $\hat{\ell}$ (yellow). The solid blue line denotes the target log-likelihood ℓ^* for a 95% confidence interval. **a** $N = 500$; **b** $N = 1000$; **c** $N = 10{,}000$

the most efficient algorithms. The errors of the three root finding methods decreased the more data became available to fit the models. However, while the binary search had a consistently low error, both the grid search and the bisection method were more prone to large errors than all other tested methods.

The algorithms developed from the constrained maximization perspective (the method by Neale and Miller and direct constrained maximization) had success rates ranging between 45% and 85% in problems with transformed covariates. In the GLM scenario, the success rate was smaller with 50 data points and higher with more data. The constrained maximization procedure was slightly more successful than the method by Neale and Miller (1997). Both methods required relatively few function evaluations, but direct constrained maximization performed better. Both methods were less prone to large errors than the grid search and the bisection method. However, the outlier-reduced error was on average more than twice as large than with any other method except RVM (Neale and Miller: 0.16, constrained maximum 0.09, RVM: 0.07).

The success of the algorithm VM depended highly on the properties of the likelihood function. In scenarios with few data and transformed covariates, VM had very low success rates (as low as 10%). When more data were added, VM became as successful as the method by Neale and Miller and direct constrained maximization. VM was highly efficient whenever results were obtained successfully. Similar to the success rate, the mean error of VM decreased strongly as more data were considered.

Wald's method had very low success rates and large errors except for the GLM with large data sets. In the models with transformed covariates, Wald's method never had a success rate above 17%.

4 Discussion

We presented an algorithm that determines the end points of profile likelihood confidence intervals both of parameters and functions of parameters with high robustness and efficiency. We tested the algorithm on a dynamical system fitted to empirical data and on benchmark problems varying in parameter number, size of the data set, and complexity of the likelihood function. In the tests, our algorithm RVM was more robust than any other considered method. At the same time, RVM was among the fastest algorithms in most scenarios. This is remarkable, because there is typically a trade-off between robustness and computational speed of optimization algorithms. RVM achieves this result by exploiting the approximately quadratic form of the log-likelihood surface in "benign" cases while maintaining high robustness with the trust-region approach. Consequently, RVM naturally extends the algorithm VM (Venzon and Moolgavkar 1988), which appeared to be highly efficient but lacking robustness in our tests.

Surprisingly, RVM turned out to be even more robust than methods based on repeated evaluations of the profile likelihood. For the bisection method and the binary search, this may be due to failures of internal optimization routines, as initial guesses far from the solution can hinder accurate convergence. The grid search method, in turn, was often aborted in the benchmark scenarios due to the limited step size, which precluded the method from identifying confidence bounds farther than 40 units away from the respective MLE. This,

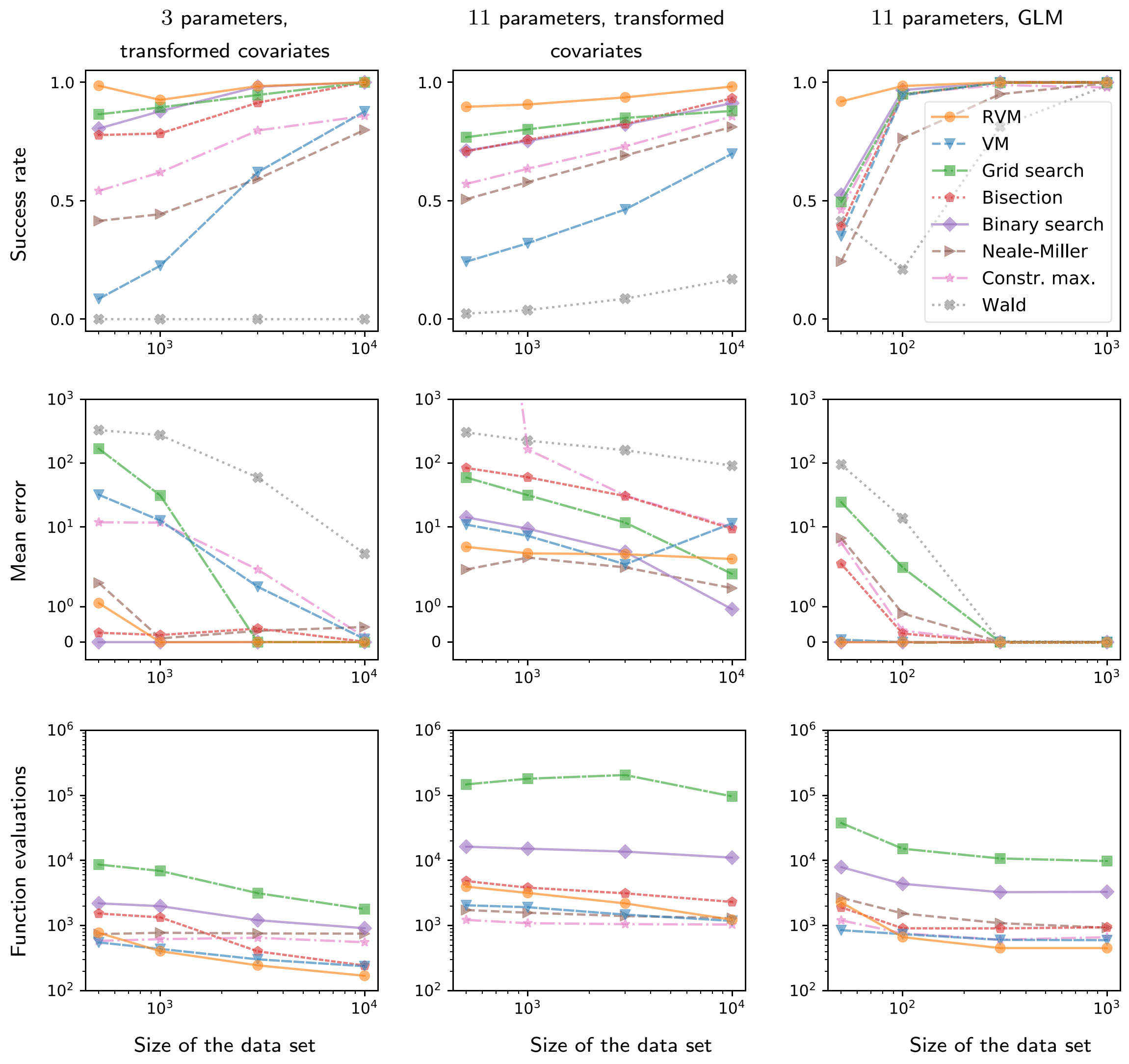

Fig. 4 Benchmark results. The success rate, the mean error, and the number of function evaluations are plotted against the size of the data set for the 3 parameter and the 11 parameter model with transformed covariates and for the 11 parameter GLM. Throughout the simulations, our algorithm RVM had the highest success rate. At the same time, RVM had a low mean error and required only few likelihood function evaluations compared to the considered alternative methods. The local minimum in the success rate for the 3 parameter model at $N = 1000$ is caused by the difficulty to compute confidence intervals for parameters that are on the verge of being estimable. The parameter values used to generate the Figures are given in Supplementary Appendix B

however, does not explain the comparatively high error in the results of the grid search, as only successful runs were considered. We therefore hypothesize that internal optimization issues were responsible for some failures.

As expected, the algorithms that searched for the confidence interval end points directly were more efficient but less robust than algorithms that repeatedly evaluate the profile likelihood. Remarkably, a "standard" algorithm for constrained optimization performed slightly better in the benchmark problems than an unconstrained optimizer operating on the modified target function suggested by Neale and Miller (1997). This indicates that the approximation introduced by Neale and Miller (1997) might not be necessary and could even be of disadvantage. For the histone H1.2 model, however, the method by Neale and Miller (1997) provided more accurate results than simple constrained maximization.

All methods implemented in this study (except RVM and VM) rely on general optimizers. Consequently, the performance of these methods depends on the chosen optimizers both in terms of computational speed and robustness. Careful adjustment of optimization parameters might make some of the implemented algorithms more efficient and thus more competitive in benchmark tests. Though we attempted to reduce potential bias by applying a variety of different methods, an exhaustive test of optimization routines was beyond the scope of this study. Nonetheless, the consistently good performance of RVM throughout our benchmark tests suggests that RVM is a good choice in many applications.

Though RVM performed well in our tests, there are instances in which the algorithm is not applicable or sufficiently efficient. This are scenarios in which (1) the log-likelihood cannot be computed directly, (2) the Hessian matrix of the log-likelihood function is hard to compute, (3) the dimension of the parameter space is very large, or (4) there are multiple points in the parameter space in which problem (4) is solved locally. Below, we briefly discuss each of these limitations.

1. In hierarchical models, the likelihood function may not be known. As RVM needs to evaluate the log-likelihood, its gradient, and its Hessian matrix, the algorithm is not applicable in these instances. Consequently, sampling-based methods, such as parametric bootstrap (Efron 1981), Monte Carlo methods (Buckland 1984), or data cloning (Ponciano et al. 2009) may then be the only feasible method to determine confidence intervals.
2. Especially in problems with a large parameter space, it is computationally expensive to compute the Hessian matrix with finite difference methods, as the number of function calls increases in quadratic order with the length of the parameter vector. Though alternative differentiation methods, such as analytical or automatic differentiation (Griewank 1989), are often applicable, there may be some instances in which finite difference methods are the only feasible alternative. In these scenarios, optimization routines that do not require knowledge of the Hessian matrix may be faster than RVM. However, higher computational speed may come with decreased robustness, and sampling-based methods might be the only remaining option if application of RVM is infeasible. Furthermore, libraries for automatic differentiation are widely available and easy to use (e.g. Albertsen et al. 2015), making RVM applicable to complex models (see Fischer et al. 2020, for such an application of RVM).
3. If the parameter space has a very high dimension (exceeding 1000), internal routines, such as inversion of the Hessian matrix, may become the dominant factor determining the speed of RVM. Though it may be possible in future to make RVM more efficient, sampling-based methods or algorithms that do not use the Hessian matrix may be better suited in these scenarios.
4. RVM as well as all other methods implemented in this study are local optimization algorithms. Therefore, the algorithms may converge to wrong results if maximization problem (4) has multiple local solutions. This is in particular the case if the confidence set $\{\theta_0 \colon \ell_{\mathrm{PL}}(\theta_0) \geq \ell^*\}$ is not connected and thus no interval. RVM reduces the issue of local extreme points by choosing steps carefully and ensuring that the point of convergence is indeed a maximum. This contrasts with VM, which could converge to the wrong confidence interval end point (e.g. maximum instead of minimum) if the initial guesses are not chosen with care. Nonetheless, stochastic optimization routines, such as genetic algorithms (Akrami et al. 2010), and sampling methods may be better suited if a local search is insufficient.

Despite these caveats, RVM is applicable to a broad class of systems. In models containing more parameters than required, such as the histone H1.2 model considered in this study, Wald's method does often not return useful results even though the results might be of scientific interest. For the histone H1.2 model, we were able to identify confidence intervals without choosing any of the reduced models with constrained parameters. The resulting confidence intervals facilitate model interpretation on a more general level than confidence intervals obtained after constraining some parameters. Though the bisection method was more efficient than RVM in the histone H1.2 example, bisection was significantly less performant in the other, broader classes of benchmark problems we considered.

Our tests showed that commonly used methods, such as VM or grid search techniques, can break down or be highly inefficient in applications—especially when inestimable parameters are present. Optimization failures are also commonly observed if not enough data are available to make the model locally approximately normal (Ren and Xia 2019). RVM is a particularly valuable tool in these instances. If RVM does not converge in time, the tuning parameters (Table 1) may be adjusted to achieve more careful step choices. Convergence issues may be caused by imprecise gradient and Hessian approximations or occur if the likelihood function is sensitive to the parameter of interest even though its confidence interval is wide. If RVM and comparable methods do not yield satisfactory results, alternative approaches, such as sampling-based methods, may need to be used.

5 Conclusion

We developed and presented an algorithm to determine profile likelihood confidence intervals. In contrast to many earlier methods, our algorithm is robust in scenarios in which lack of data or a complicated likelihood function make it difficult to find the bounds of profile likelihood confidence intervals. In particular, our methods is applicable in instances in which parameters are not estimable and in cases in which the likelihood function has strong nonlinearities. At the same time, our method efficiently exploits the local asymptotic normality of models if enough data are available.

We tested our method on an empirical dataset and benchmark problems with different difficulty. Throughout these problem sets, our method was the most robust while also being among the fastest algorithms. We therefore believe that

RVM can be helpful to researchers and modelers across disciplines.

Acknowledgements The authors would like to thank Carlos Contreras for his assistance with the histone H1.2 model and data set. Furthermore, the authors would like to give thanks to the members of the Lewis Research Group at the University of Alberta for helpful feedback and discussions.

Author contributions Samuel M. Fischer and Mark A. Lewis jointly conceived the project; Samuel M. Fischer conceived the algorithm, conducted the mathematical analysis, implemented the algorithm, and wrote the manuscript. Mark A. Lewis revised the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Code availability A Python implementation of the described algorithm and the test procedures can be retrieved from the python package index as package *ci-rvm* (see https://pypi.org/project/ci-rvm).

Declarations

Competing interests The authors declare no competing interests.

References

Akrami, Y., Scott, P., Edsjö, J., Conrad, J., Bergström, L.: A profile likelihood analysis of the constrained MSSM with genetic algorithms. J. High Energy Phys. **2010**(4), 57 (2010)

Albertsen, C.M., Whoriskey, K., Yurkowski, D., Nielsen, A., Flemming, J.M.: Fast fitting of non-Gaussian state-space models to animal movement data via Template Model Builder. Ecology **96**(10), 2598–2604 (2015)

Brodtkorb, P.A., D'Errico, J.: numdifftools 0.9.39. Retrieved from https://github.com/pbrod/numdifftools (2019)

Buckland, S.T.: Monte Carlo confidence intervals. Biometrics **40**(3), 811 (1984)

Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. MPS-SIAM series on optimization. Society for Industrial and Applied Mathematics, Philadelphia (2000)

Contreras, C., Villasana, M., Hendzel, M.J., Carrero, G.: Using a model comparison approach to describe the assembly pathway for histone H1. PLOS ONE **13**(1), e0191562 (2018)

Cook, R.D., Weisberg, S.: Confidence curves in nonlinear regression. J. Am. Stat. Assoc. **85**(410), 544–551 (1990)

Cox, D.R., Snell, E.J.: Analysis of Binary Data. Number 32 in Monographs on Statistics and Applied Probability, 2nd edn. Routledge, Boca Raton (1989)

DiCiccio, T.J., Tibshirani, R.: On the Implementation of Profile Likelihood Methods. Technical report, University of Toronto, Department of Statistics (1991)

Efron, B.: Nonparametric standard errors and confidence intervals. Can. J. Stat. **9**(2), 139–158 (1981)

Eubank, R.L., Webster, J.T.: The singular-value decomposition as a tool for solving estimability problems. Am. Stat. **39**(1), 64 (1985)

Fischer, S.M., Beck, M., Herborg, L.-M., Lewis, M.A.: A hybrid gravity and route choice model to assess vector traffic in large-scale road networks. R. Soc. Open Sci., 1–26 (2020)

Gardner, W., Mulvey, E.P., Shaw, E.C.: Regression analyses of counts and rates: Poisson, overdispersed Poisson, and negative binomial models. Psychol. Bull. **118**(3), 392–404 (1995)

Gimenez, O., Choquet, R., Lamor, L., Scofield, P., Fletcher, D., Lebreton, J.-D., Pradel, R.: Efficient profile-likelihood confidence intervals for capture-recapture models. J. Agric. Biol. Environ. Stat. **10**(2), 184–196 (2005)

Griewank, A.: On automatic differentiation. Math. Programm. Recent Dev. Appl. **6**(6), 83–107 (1989)

Jones, E., Oliphant, T., Peterson, P.: SciPy: open source scientific tools for Python. Retrieved from (2001). https://scipy.org/

Kraft, D.: A Software Package for Sequential Quadratic Programming. Technical Report DFVLR-FB 88-28, DLR German Aerospace Center—Institute for Flight Mechanics, Köln (1988)

Lai, K.-L., Crassidis, J., Cheng, Y., Kim, J.: New complex-step derivative approximations with application to second-order Kalman filtering. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California. American Institute of Aeronautics and Astronautics (2005)

Lalee, M., Nocedal, J., Plantenga, T.: On the implementation of an algorithm for large-scale equality constrained optimization. SIAM J. Optim. **8**(3), 682–706 (1998)

Moerbeek, M., Piersma, A.H., Slob, W.: A comparison of three methods for calculating confidence intervals for the benchmark dose. Risk Anal. **24**(1), 31–40 (2004)

Neale, M.C., Miller, M.B.: The use of likelihood-based confidence intervals in genetic models. Behav. Genet. **27**(2), 113–120 (1997)

Nocedal, J., Wright, S.J.: Numerical Optimization. Springer series in operations research, 2nd edn. Springer, New York (2006)

Pek, J., Wu, H.: Profile likelihood-based confidence intervals and regions for structural equation models. Psychometrika **80**(4), 1123–1145 (2015)

Penrose, R.: A generalized inverse for matrices. Math. Proc. Camb. Philos. Soc. **51**(3), 406–413 (1955)

Ponciano, J.M., Taper, M.L., Dennis, B., Lele, S.R.: Hierarchical models in ecology: confidence intervals, hypothesis testing, and model selection using data cloning. Ecology **90**(2), 356–362 (2009)

Raghuram, N., Carrero, G., Stasevich, T.J., McNally, J.G., Th'ng, J., Hendzel, M.J.: Core histone hyperacetylation impacts cooperative behavior and high-affinity binding of histone H1 to chromatin. Biochemistry **49**(21), 4420–4431 (2010)

Ramsay, J.O., Hooker, G., Campbell, D., Cao, J.: Parameter estimation for differential equations: a generalized smoothing approach: parameter estimation for differential equations. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **69**(5), 741–796 (2007)

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., Timmer, J.: Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics **25**(15), 1923–1929 (2009)

Ren, X., Xia, J.: An algorithm for computing profile likelihood based pointwise confidence intervals for nonlinear dose-response models. PLOS ONE **14**(1), e0210953 (2019)

Stryhn, H., Christensen, J.: Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology. Vina del Mar (2003)

Venzon, D.J., Moolgavkar, S.H.: A method for computing profile-likelihood-based confidence intervals. Appl. Stat. **37**(1), 87 (1988)

Viallefont, A., Lebreton, J.-D., Reboulet, A.-M., Gory, G.: Parameter identifiability and model selection in capture-recapture models: a numerical approach. Biometr. J. J. Math. Methods Biosci. **40**(3), 313–325 (1998)

Wu, H., Neale, M.C.: Adjusted confidence intervals for a bounded parameter. Behav. Genet. **42**(6), 886–898 (2012)

Yuan, Y.-X.: Recent advances in trust region algorithms. Math. Program. **151**(1), 249–281 (2015)