A comparison of optimization solvers for log binomial regression including conic programming

Schwendinger, Florian; Grün, Bettina; Hornik, Kurt

Published in:
Computational Statistics

DOI:
10.1007/s00180-021-01084-5

Published: 01/01/2021

Document Version
Publisher's PDF, also known as Version of record

Computational Statistics (2021) 36:1721–1754
https://doi.org/10.1007/s00180-021-01084-5

ORIGINAL PAPER

A comparison of optimization solvers for log binomial regression including conic programming

Florian Schwendinger[1] · **Bettina Grün**[1] · **Kurt Hornik**[1]

Received: 25 February 2020 / Accepted: 30 January 2021 / Published online: 22 February 2021

Abstract
Relative risks are estimated to assess associations and effects due to their ease of interpretability, e.g., in epidemiological studies. Fitting log-binomial regression models allows to use the estimated regression coefficients to directly infer the relative risks. The estimation of these models, however, is complicated because of the constraints which have to be imposed on the parameter space. In this paper we systematically compare different optimization algorithms to obtain the maximum likelihood estimates for the regression coefficients in log-binomial regression. We first establish under which conditions the maximum likelihood estimates are guaranteed to be finite and unique, which allows to identify and exclude problematic cases. In simulation studies using artificial data we compare the performance of different optimizers including solvers based on the augmented Lagrangian method, interior-point methods including a conic optimizer, majorize-minimize algorithms, iteratively reweighted least squares and expectation-maximization algorithm variants. We demonstrate that conic optimizers emerge as the preferred choice due to their reliability, lack of requirement to tune hyperparameters and speed.

Keywords Log-binomial regression · Relative risk · Optimization · Conic programming

1 Introduction

Regression models for binary outcomes are commonly used to either estimate odds ratios or relative risks. Since odds ratios are commonly misinterpreted by non-statisticians, several authors (e.g., Davies et al. 1998; Holcomb et al. 2001), especially in epidemiological studies, suggested that relative risks should preferably be estimated and reported. Odds ratios are obtained from logistic regression, which makes use of

✉ Florian Schwendinger
FlorianSchwendinger@gmx.at

[1] Wirtschaftsuniversität Wien, Wien, Austria

the logit link, the canonical link function for binary responses in the generalized linear model (GLM) framework. The logit link ensures that the estimated probabilities are bounded between 0 and 1. Relative risks are estimated by log-binomial regression, where the logit link is replaced by a log link. The log link does not guarantee that the estimated probabilities are bounded between 0 and 1. Therefore, the parameter space needs to be constrained in order to obtain valid probability estimates.

If the log-binomial regression model is fitted using the iteratively reweighted least squares (IRLS) method, i.e., the standard algorithm for fitting GLMs, step-halving is used to ensure that the parameter estimates are within the restricted parameter space. However, several authors noted that this procedure may fail to converge (e.g., De Andrade and Carabin 2011; Williamson et al. 2013). To overcome this issue, alternative methods to estimate the log-binomial regression model were proposed in the literature (for an overview, see, for example, Lumley et al. 2006). More recently, Luo et al. (2014) and de Andrade and de Leon Andrade (2018) suggest to reformulate the model estimation problem as an optimization problem with linear constraints. Both studies use the `constrOptim` function from the R package **stats** (Core Team 2020), which implements an adaptive version of the log-barrier method (Lange 1994). A drawback of using `constrOptim`, however, is that convergence to the optimum depends on the choice of the starting values and other tuning parameters.

In this study we systematically compare the performance of a range of different optimizers for estimating the log-binomial regression model and relative risks. In particular we explore the possible advantages of using modern conic optimization solvers. Linear optimization solvers like GLPK (GNU Linear Programming Kit; Makhorin 2011) are known to reliably recover a global solution. Due to recent advances in conic programming and due to the availability of new conic optimization solvers, e.g., ECOS (Embedded Conic Solver; Domahidi et al. 2013), it is now possible to recover global solutions to many nonlinear convex optimization problems with a similar reliability than for linear optimization problem. This nonlinear convex optimization includes problems where the objective and constraints are comprised of linear, quadratic, logarithmic, exponential and power terms. From a user perspective the three main advantages of these solvers are: (1) similarly to linear optimization solvers, there is no need to provide starting values; (2) fewer tuning parameters (e.g., the barrier parameter) have to be specified, since they are calculated internally; (3) the returned status codes, which signal whether an acceptable solution was found or an error occurred, are more reliable than for general nonlinear optimization solvers.

The rest of the paper is structured as follows. In Sect. 2 we give a definition of the log-binomial model and clarify in which cases the maximum likelihood estimate (MLE) is unique and finite. In Sect. 3 we review the different optimization solvers used in the following comparison. In Sect. 4 we assess the reliability, accuracy and speed of the solvers on different simulation settings. Section 5 concludes the paper.

2 Log-binomial regression

2.1 Model specification

The log-binomial regression model (LBRM) is a generalized linear model (GLM) with binary response and log link. Let $y \in \{0, 1\}^n$ be a binary response variable and $X \in \mathbb{R}^{n \times p}$ the model matrix built from the k covariates plus an intercept ($p = k + 1$). Then the log-binomial model assumes that the probability of the dependent variable being equal to 1 given the covariates for observation i is:

$$\log(P(Y = 1|X_{i*})) = X_{i*}\beta, \tag{1}$$

where X_{i*} refers to the i-th row of the model matrix X.

Exponentiating the coefficient vector β directly gives the adjusted relative risk (RR). Specifically the RR of X_{i*} compared to X_{j*} is given by

$$\text{RR} = \frac{P(Y = 1|X_{i*})}{P(Y = 1|X_{j*})} = \frac{\exp(X_{i*}\beta)}{\exp(X_{j*}\beta)} = \exp(\beta_k), \tag{2}$$

if X_{i*} and X_{j*} only differ with respect to the k-th covariate in the way that $X_{ik} = X_{jk} + 1$.

To obtain the MLE of β the following optimization problem has to be solved:

$$\begin{aligned} &\underset{\beta}{\text{maximize}}\ \ell(\beta) = \sum_{i=1}^{n} y_i\ X_{i*}\beta + (1 - y_i)\ \log(1 - \exp(X_{i*}\beta)) \\ &\text{subject to } X\beta \leq 0. \end{aligned} \tag{3}$$

The constraint $X\beta \leq 0$ is necessary to ensure $P(Y = 1|X_{i*}) = \exp(X_{i*}\beta) \leq 1,\ \forall i = 1, \ldots, n$. Let X^0 be the submatrix of X obtained by keeping only the rows $I^0 = \{i|y_i = 0\}$ and X^1 the submatrix obtained by keeping only the rows $I^1 = \{i|y_i = 1\}$. Then the effective domain of ℓ is (Calafiore and Ghaoui 2014) is

$$\text{dom}\ \ell = \{\beta| - \infty < X_{i*}\beta < 0\ \forall i \in I^0\} \cap \{\beta| - \infty < X_{i*}\beta \leq 0\ \forall i \in I^1\}.$$

The gradient of the log-likelihood is

$$\nabla\ell(\beta) = \sum_{i \in I^1} X_{i*}^\top - \sum_{i \in I^0} \frac{X_{i*}^\top \exp(X_{i*}\beta)}{1 - \exp(X_{i*}\beta)} \tag{4}$$

and the Hessian is

$$\nabla^2\ell(\beta) = -\sum_{i \in I^0} X_{i*}^\top X_{i*} \frac{\exp(X_{i*}\beta)}{(1 - \exp(X_{i*}\beta))^2}\ . \tag{5}$$

2.2 Properties of the LBRM

Before determining the MLE, in particular using numerical optimization methods, it is important to assess if the optimization problem has a finite and unique solution. We say the MLE is finite, if all its elements $\hat{\beta}_i$ for all $i \in I$ are finite. In the following we establish conditions for the uniqueness and finiteness of the MLE, accompanied by intuitive proofs which reveal interesting insights into the structure of the problem. For a more general treatment of this topic, we refer to Kaufmann (1988), who establishes conditions for the finiteness of the MLE for quantal and ordinal response models.

2.2.1 Uniqueness of the MLE

In order for the solution to be unique, it suffices that the negative log-likelihood is strictly convex (see, e.g., Calafiore and Ghaoui 2014, page 255). Since the negative log-likelihood is twice differentiable, we can prove the strict convexity of $-\ell(\beta)$ by showing that the Hessian is positive definite for all $\beta \in \operatorname{dom} \ell$ (Calafiore and Ghaoui 2014, p. 236). Rewriting the Hessian of the negative log-likelihood as

$$-(\nabla^2 \ell(\beta)) = (D^{\frac{1}{2}} X^0)^\top (D^{\frac{1}{2}} X^0) \tag{6}$$

with D the diagonal matrix, where the (i,i)-entry is equal to $\frac{\exp(X_{i*}\beta)}{(1-\exp(X_{i*}\beta))^2}$, reveals that the Hessian of the LBRM is a Gram matrix. Gram matrices are always positive semidefinite. Furthermore, a Gram matrix $G = A^\top A$, $A \in \mathbb{R}^{m \times n}$ is positive definite if and only if the columns of A are linearly independent (see, e.g., Horn and Johnson 2012, p. 441).

Theorem 1 *If X^0 has full column rank, then the MLE of the LBRM is unique.*

Proof Assuming that X^0 has full column rank, since $\frac{\exp(X_{i*}\beta)}{(1-\exp(X_{i*}\beta))^2} > 0$ for all $i \in I^0, \beta \in \operatorname{dom} \ell$ it follows that $(D^{\frac{1}{2}} X^0)$ has full column rank and therefore that the Hessian is positive definite for all $\beta \in \operatorname{dom} \ell$. □

To complement these results note that de Andrade and de Leon Andrade (2018) also present two sets of sufficient conditions for the MLE of an LBRM to be unique. In particular their first set of sufficient conditions requires X to have full column rank and at least one failure to be observed and their second set requires that failures as well as successes are observed and both X^0 and X have full column rank. More generally the results in Kaufmann (1988) imply that the MLE of the LBRM is unique if and only if X has full column rank.

2.2.2 Finiteness of the MLE

Silvapulle (1981) gives necessary and sufficient criteria for the uniqueness and finiteness of the MLE for binomial response models with unrestricted parameter space. The criteria imply that a certain degree of overlap between the covariates of the successes and failures is needed. Albert and Anderson (1984) classify the overlap between

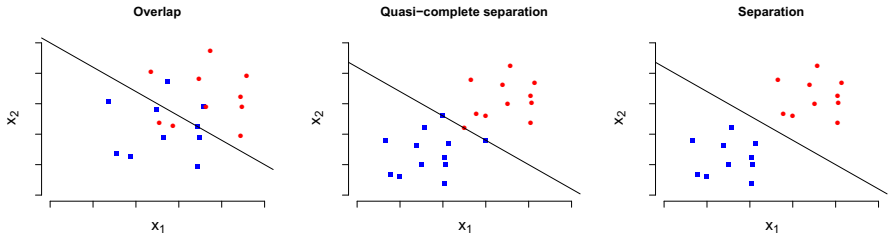

Fig. 1 Illustration of overlap, quasi-complete separation and separation

the covariates into three different settings: *complete separation*, *quasi-complete separation* and *overlap*. Furthermore, they show that for logistic regression overlap is necessary and sufficient for the finiteness of the MLE. A model matrix X is said to exhibit complete separation if there exists a β such that $X^0\beta < 0$ and $X^1\beta > 0$. Similarly, X is said to exhibit quasi-complete separation if there exists a $\beta \neq 0$ such that $X^0\beta \leq 0$ and $X^1\beta \geq 0$. For an illustration of these three settings see Fig. 1.

However, due to the restricted parameter space $\{\beta \in \mathbb{R}^p | X\beta \leq 0\}$, the results of Silvapulle (1981) and Albert and Anderson (1984) do not apply directly for the LBRM. In particular, overlap is only a sufficient, but not a necessary condition for the finiteness of the MLE. Kaufmann (1988) shows that for binomial response models with finite upper bound (e.g., the LBRM) the recession cone of ℓ is given by

$$R = \{\beta | X^0\beta \leq 0 \wedge X^1\beta = 0\}. \tag{7}$$

Therefore, assuming that X has full column rank, the MLE $\hat{\beta}$ is finite and unique if and only if $\{\beta | X^0\beta \leq 0 \wedge X^1\beta = 0\} = \{0\}$. This can be translated into the following sufficient condition. The MLE $\hat{\beta}$ is finite and unique if X^1 has full column rank or if X has full column rank and the covariates overlap.

2.2.3 Detecting infinite components of the MLE

If the optimization task could be performed up to an arbitrary precision, the infinite components of the MLE could be directly observed as an output of the optimization step. However, since all applicable numerical optimization solvers do not provide arbitrary precision, in the case of the LBRM the optimizer will always return a finite solution (or in some rare cases an error). In fact it is not even guaranteed that the magnitude of the obtained coefficients corresponding to the infinite components is high. It is thus necessary to check the finiteness of the MLE components before their estimation.

Konis (2007) surveys separation detection methods for logistic regression and suggests using a linear programming approach to detect separation. This approach is implemented in the R package **safeBinaryRegression** (Konis and Fokianos 2009),

Table 1 Example with finite MLE of the LBRM

X_{*1}	X_{*2}	X_{*3}	y
1	1	2	0
1	0	1	0
1	3	1	0
1	2	4	1
1	3	6	1
1	4	8	1

where the following linear program is solved.

$$\begin{array}{ll} \underset{\beta}{\text{maximize}} & \sum_{i \in I^1} X_{i*}\beta - \sum_{i \in I^0} X_{i*}\beta \\ \text{subject to} & X^0\beta \leq 0 \\ & X^1\beta \geq 0 \end{array} \tag{8}$$

The solution of the linear program (8) is either the zero vector, in which case the data is overlapped, or unbounded, in which case the data is (quasi-)separated.

Since for the LBRM overlap of the data points is only a sufficient, but not a necessary condition for the finiteness of the MLE, this approach cannot be directly employed to detect all cases which yield a finite MLE. In fact, there even exist data configurations where neither X^1 has full column rank, nor the data points are overlapped, but nevertheless the MLE is finite. Table 1 provides an example: clearly, X^1 does not have full column rank and the data is separated. Nevertheless the solution $\hat{\beta} = (-1.645, -0.446, 0.429)$ has no infinite component.

In fact, taking into account the recession cone from Eq. (7) the linear programming approach from Eq. (8) can be modified to verify the finiteness of the MLE in the LBRM context. Consider the following linear program:

$$\begin{array}{ll} \underset{\beta}{\text{maximize}} & -\sum_{i \in I^0} X_{i*}\beta \\ \text{subject to} & X^0\beta \leq 0 \\ & X^1\beta = 0. \end{array} \tag{9}$$

The MLE has only finite components if the solution of this linear program is a zero vector. If the MLE contains infinite components, the linear programming problem is unbounded. A function to diagnose the properties of the LBRM, called `diagnose_lbrm`, can be found in the appendix.

Note that the roles of failures and successes are not interchangeable for the LBRM. Their role is symmetric for the logistic binomial regression model where successes and failures may be interchanged and $-\beta$ of the original formulation corresponds to the regression coefficients of the interchanged formulation. For the LBRM no correspondence between the constraints imposed on the regression coefficients can be established if the roles of failures and successes are interchanged. This implies that infinite components of the MLE need to be detected conditional on having fixed the role of failures and successes.

3 Optimization methods and solvers

We use general purpose optimizers as well as specialized algorithms and implementations to systematically compare the performance of different optimization methods and solvers for the LBRM. As general-purpose estimators we consider the optimization solvers `auglag` (Varadhan 2015), `constrOptim`, ECOS and IPOPT (Wächter and Biegler 2006). The special purpose implementations for the LBRM considered are the standard implementation of the IRLS algorithm in function `glm` from the base R package **stats** as well as the improved version available in the R package **glm2** (Marschner 2011). Specific variants of an expectation-maximization (EM) algorithm developed for the LBRM and implemented in the R package **logbin** (Donoghoe and Marschner 2018) are also employed.

The general purpose optimizers were carefully selected to differ considerably from each other, by either the method or the implementation. The solver `constrOptim` implements a special case of the majorize-minimize (MM) algorithm and has emerged in the literature as one of the preferred general purpose optimizers to estimate the LBRM. `auglag` implements the augmented Lagrangian method and is a popular solver among statisticians. ECOS as well as IPOPT implement interior point methods. However, there are considerable differences between the two implementations. ECOS implements an interior point method based on modern conic optimization. We included ECOS in the comparison in particular because of its very reliable return codes. This means in the context of the LBRM, under the assumption that the MLE is finite and unique, if ECOS signals success we can be almost certain that the global maximum was returned. IPOPT was selected since it is a popular solver for nonlinear optimization problems in the optimization literature. In the following we provide some details on the algorithms of the selected solvers and their implementations.

3.1 Augmented Lagrangian method

Instead of solving the constrained problem from Eq. (3) directly the augmented Lagrangian method solves a sequence of unconstrained optimization problems of the form:

$$\underset{\beta}{\text{minimize}} \left\{ -\ell(\beta) - \sum_{i=1}^{n} \lambda_i d_i(\beta, \lambda_i, \sigma) + \frac{\sigma}{2} \sum_{i=1}^{n} d_i(\beta, \lambda_i, \sigma)^2 \right\}. \tag{10}$$

The multipliers λ_i and the penalty parameter σ are updated in the outer iterations and d_i are the modified inequality constraints

$$d_i(\beta, \lambda_i, \sigma) = \begin{cases} -X_{i*}\beta & \text{if } -X_{i*}\beta \leq \frac{\lambda_i}{\sigma} \\ \frac{\lambda_i}{\sigma} & \text{otherwise.} \end{cases} \tag{11}$$

In the outer iterations the problem is transformed from the form shown in Eq. (3) to the form given in Eq. (10). In the inner iterations the transformed problem is solved

with a nonlinear unconstrained optimization solver. More details on the augmented Lagrangian method can be found in, e.g., Madsen et al. (2004).

The `auglag` solver from the **alabama** package (Varadhan 2015) implements the augmented Lagrangian method in R. It allows to choose between several algorithms for solving the transformed nonlinear unconstrained optimization problem in the inner iterations. By default it uses the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm from the `optim` solver included in the **stats** package. Using `auglag` with BFGS the objective and the constraint functions should be at least once differentiable functions and preferably twice differentiable functions.

3.2 Interior-point methods

Interior-point (or barrier) methods were originally developed for nonlinear optimization in the 1960s (see, e.g., Fiacco and McCormick 1968) and became popular again in the 1980s, when Karmarkar (1984) introduced a polynomial-time interior-point method for linear programming. Today almost all linear programming solvers implement either interior-point methods or the simplex algorithm. Additionally interior-point methods play an important role in large-scale nonlinear programming. The basic principle of interior-point methods is to approximate the constrained optimization problem:

$$\underset{x}{\text{minimize}} \left\{ f(x) \text{ subject to } g_i(x) \leq 0, i = 1, \ldots, n \right\}, \tag{12}$$

by a series of unconstrained problems of the form:

$$\underset{x}{\text{minimize}} \left\{ f(x) - \mu \sum_{i=1}^{n} \psi(-g_i(x)) \right\}, \tag{13}$$

where $\mu > 0$ is the penalty parameter and $\psi()$ is a suitable barrier. A typical barrier choice for the nonnegative half-axis is $\psi(s) = \log(s)$ (Nesterov and Nemirovskii 1994). More recent approaches slightly deviate from the basic principle introduced above. For example, modern interior-point solvers use a problem formulation with slack variables. This has the advantage that the initial value is not required to lie within the feasible region (Nocedal and Wright 2006).

3.2.1 Conic optimization with the ECOS solver

Recent advances in conic programming (Chares 2009; Serrano 2015), specifically the availability of solvers for the exponential cone, make it possible to solve the LBRM by the means of conic programming. The conic program can be written as

$$\underset{x}{\text{minimize}} \{ a_0^\top x : Ax + s = b, s \in \mathcal{K} \}, \tag{14}$$

where $a_0 \in \mathbb{R}^p$ is the coefficient vector of the objective function, $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix of the constraints, $b \in \mathbb{R}^m$ is the right-hand side of the constraints

and $\mathcal{K}$ a nonempty closed convex cone (see, e.g., Nemirovski 2006). In the case, where $\mathcal{K}$ is the Cartesian product of linear cones $\mathcal{K}_{\text{lin}} = \{x \in \mathbb{R} | x \geq 0\}$, Eq. (14) reduces to a linear optimization problem in standard form:

$$\underset{x}{\text{minimize}}\{a_0^\top x : Ax \leq b\}. \tag{15}$$

In theory any convex nonlinear optimization problem can be expressed in terms of Eq. (14). Practically the kind of optimization problems, which can be solved with conic programming, is limited by the cones supported by the selected optimization solver. Serrano (2015) extends the primal-dual interior-point algorithm of the ECOS solver to solve problems with the exponential cone

$$\mathcal{K}_{\text{exp}} = \{(x, y, z) \in \mathbb{R}^3 \mid y > 0,\ ye^{\frac{x}{y}} \leq z\} \cup \{(x, 0, z) \in \mathbb{R}^3 \mid x \leq 0,\ z \geq 0\}.$$

This extension makes it possible to employ ECOS for solving convex optimization problems where the objective and/or the constraints contain logarithmic and/or exponential terms. The main distinction of interior-point methods for conic programming to general interior-point methods is that the constraint functions g_i are all linear and slack variables s_i are introduced to obtain equality constraints. Nonlinear constraint functions in the original problem formulations are accounted for by restricting the slack variables s_i to a certain cone, e.g., the log barrier is used for linear inequality constraints. This reformulation implies that conic solvers require as input not functions but, instead, matrices and vectors, which facilitates scaling and to make use of efficient barrier terms. For more technical details about the ECOS solver we refer to Domahidi et al. (2013) and Domahidi (2013).

The introduction of the exponential cone allows the LBRM model from Eq. (3) to be rewritten as a conic optimization problem:

$$\begin{aligned} \underset{(\beta,\gamma,\delta)}{\text{maximize}}\ & (y^\top X)\beta + \mathbf{1}^\top \delta \\ \text{subject to}\ & X_{i*}\beta \leq 0 \text{ for all } i \in I^1 \\ & (X_{i*}\beta, 1, \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in I^0 \\ & (\delta_i, 1, 1 - \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in I^0 \\ & \beta \in \mathbb{R}^p, \gamma \in \mathbb{R}^{n_0}, \delta \in \mathbb{R}^{n_0} \end{aligned} \tag{16}$$

where $\mathbf{1}$ refers to the vector of ones. See Boyd and Vandenberghe (2004) for an introduction into conic programming and Theußl et al. (2020) for details on how to derive the representation needed by conic solvers.

We use the following simple example to explain the steps taken to transform the problem stated in Eq. (3) into its conic form. Assume the following optimization problem:

$$\begin{aligned} \underset{\beta}{\text{maximize}}\ & \log(1 - \exp(2\beta)) \\ \text{subject to}\ & 2\beta \leq 0 \end{aligned} \tag{17}$$

This problem can be rewritten into its epigraph form (see, e.g., Boyd and Vandenberghe 2004):

$$\begin{aligned} &\underset{(\beta,\delta)}{\text{maximize}} \ \delta \\ &\text{subject to } \log(1 - \exp(2\beta)) \geq \delta \\ &\qquad\qquad 2\beta \leq 0 \end{aligned} \tag{18}$$

The constraint $\log(1 - \gamma) \geq \delta$ is equivalent to $\exp(\delta) \leq 1 - \gamma$. This can also be expressed by the conic constraint $(\delta, 1, 1 - \gamma) \in \mathcal{K}_{\text{exp}}$. Similarly the constraint $\exp(2\beta) \leq \gamma$ can be expressed by the conic constraint $(2\beta, 1, \gamma) \in \mathcal{K}_{\text{exp}}$. With these reformulation techniques, we can rewrite the problem stated in Eq. (3) into its conic form stated in Eq. (16).

In addition to ECOS this problem may also be solved using other convex optimization solvers such as SCS (O'Donoghue 2015; O'Donoghue et al. 2016) and MOSEK (MOSEK 2017). We used the R optimization infrastructure package **ROI** (Hornik et al. 2020) to formulate the conic optimization problem. Additional details on how to rewrite the original LBRM problem into the form needed by a conic optimization solver are given in Theußl et al. (2020) and on the **ROI** homepage.[1]

3.2.2 IPOPT

The IPOPT solver (Wächter and Biegler 2006; Wächter 2009) implements an interior-point line-search filter method and is designed to be used for large-scale convex and non-convex nonlinear optimization problems. In the case of the LBRM, IPOPT solves the following optimization problem:

$$\begin{aligned} &\underset{(\beta,s)}{\text{minimize}} \quad -\ell(\beta) - \mu \textstyle\sum_{i=1}^{n} \log(s_i) \\ &\text{subject to} \quad X_{i*}\beta + s_i = 0 \qquad\qquad i = 1, \ldots, n. \end{aligned} \tag{19}$$

The barrier term $\mu \sum_{i=1}^{n} \log(s_i)$ ensures that $s_i > 0$ for all $i = 1, \ldots, n$. IPOPT offers different options for the update of the barrier parameter μ; the default is the monotone Fiacco–McCormick strategy (Fiacco and McCormick 1968). Wächter (2009) points out that the objective and constraint functions should be at least once differentiable functions and preferably twice differentiable functions.

3.3 MM algorithm

Lange (2016) points out that the MM algorithm (or MM principle; Ortega and Rheinboldt 1970; De Leeuw and Heiser 1977) strictly speaking is not an algorithm, but a technique for generating optimization algorithms. The basic principle is based on the replacement of a hard optimization problem by a sequence of easier optimization problems. Thereby the original objective function is replaced by a majorizing surrogate function. The surrogate function depends on the current iterate and is updated in

[1] http://roi.r-forge.r-project.org/.

each iteration. Several important optimization algorithms can be seen as a special case of the MM-algorithm (e.g., gradient descent, EM algorithm, IRLS algorithm, …).

Current LBRM implementations are often based on `constrOptim` which implements an adaptive barrier method as described in Lange (1994). Originally this adaptive barrier method was presented as a combination of ideas from interior point methods and from the EM algorithm. In more recent work, Hunter and Lange (2004) present the adaptive barrier method as a special case of the MM algorithm. In the following we outline the algorithm implemented in `constrOptim` for the special case of the LBRM. The problem from Eq. (3) is solved by iteratively solving the following unconstrained optimization problem:

$$\underset{\beta}{\text{minimize}} \left\{ -\ell(\beta) - \mu \sum_{i=1}^{n} \left(c_i^{(k)} \log(-X_{i*}\beta) + X_{i*}\beta \right) \right\}, \tag{20}$$

where $c_i^{(k)} = -X_{i*}\beta^{(k)}$ and $\beta^{(k)}$ refers to the minimum obtained in the previous iteration. The barrier parameter μ is fixed and can be provided by the user. Any point in the interior of the feasible region can be used as a starting value $\beta^{(0)}$. `constrOptim` assumes that the objective function is convex and using `constrOptim` in combination with BFGS, the objective function should be at least once, and preferably twice, differentiable.

3.4 Special purpose solvers

A special purpose solver is especially developed to solve a specific optimization problem and, hence, the objective function and the constraints cannot be altered by the user. To the best of our knowledge, there exist four different special purpose solvers for log-binomial regression in R, namely `glm`, `glm2` and two EM algorithms implemented in the **logbin** package.

3.4.1 `glm/glm2`

The IRLS algorithm is the default algorithm for maximum likelihood estimation of GLMs in many statistical software environments. At its core, it solves a weighted least squares problem in every iteration. Most implementations use step-halving to prevent overshooting and to keep the estimates within a potentially restricted parameter space. The IRLS algorithm is widely used because it is very fast and it reliably delivers the solution for many commonly fitted GLMs. However, for LBRM, Marschner (2011) reports that the `glm` function from the **stats** package in some cases fails to converge and suggests an alternative implementation available in the **glm2** package which fixes some of the convergence issues by imposing a stricter step-halving criterion.

3.4.2 logbin

The **logbin** package bundles methods for log-binomial regression. Among the available fitting methods are two EM algorithm implementations. The combinatorial EM

(available by choosing the method `"cem"`) estimates the log-binomial likelihood on the restricted parameter space, by solving a family of EM algorithms and then choosing the best of the obtained results. The second implementation (available by choosing the method `"em"`) uses a parameter extension strategy. For more details, we refer to Donoghoe and Marschner (2018).

4 Solver comparison

The aim of the simulation studies is to assess the weaknesses and strengths of the selected solvers in the context of the LBRM. We evaluate the performance of the solvers on three different simulation tasks. Simulation tasks A and B are commonly used in the RR literature. Including these simulations allows to compare our results, where additional solvers are included in the simulation studies, to those obtained in previous studies. In simulation task C we define more challenging simulation settings, especially designed to assess the speed and the reliability of the different solvers.

For all three simulation tasks we repeat each simulation scenario 1000 times with 500 observations each. The simulations were performed using machines with 20 cores of Intel Xeon E5-20650v3 2.30 GHz and 256 GB RAM. However, since we developed the simulations on a 4 core Intel i5-3320M machine with 8 GM RAM, we can confirm that the simulation studies would also run with much less resources. The code to reproduce the simulations and estimation is publicly available at https://r-forge.r-project.org/projects/lbrm/.

4.1 Performance and evaluation criteria

For all three simulation tasks the following performance criteria are determined and compared:

- Non-convergence rate (NCR): relative number of times the solver signaled non-convergence or issues with the returned solution.
- Absolute log-likelihood difference (ALLD): average absolute difference between the log-likelihood obtained by the solver and the highest log-likelihood obtained by all solvers $\ell(\hat{\beta}^*)$:

$$\text{ALLD} = \frac{1}{1000} \sum_{k=1}^{1000} |\ell(\hat{\beta}_k) - \ell(\hat{\beta}_k^*)|.$$

For simulation tasks A and B we also report for the RR parameter of interest:

- Bias (BIAS): relative bias in percent determined by

$$\text{BIAS} = \frac{100}{1000} \sum_{k=1}^{1000} \frac{\widehat{\text{RR}}_k - \text{RR}}{\text{RR}}.$$

– Root mean squared error (RMSE): determined by

$$\text{RMSE} = \sqrt{\frac{1}{1000}\sum_{k=1}^{1000}\left(\log(\widehat{\text{RR}}_k) - \log(\text{RR})\right)^2}.$$

– Coverage rate (CR): the percentage of confidence intervals covering the true parameter (at a 95% confidence level). Since in the presence of binding constraints using the inverse of the observed information matrix as estimator of the covariance matrix (Blizzard and Hosmer 2006) is no longer valid, we used the estimator suggested in de Andrade and de Leon Andrade (2018) whenever at least one of the constraints was binding.

For the more challenging simulation task C, we also include timings and a speed comparison.

4.2 Solver settings

Results obtained when comparing optimization solvers used for the LBRM do no only depend on the simulation scenarios used, but also on a number of specific choices made when applying the solvers. In particular the solution reported by a solver may depend on the choice of the tuning parameters of the algorithm, the starting value, and the specific implementation of the objective/gradient functions. To provide the reader with a better understanding of the strengths and weaknesses of each solver we report for each solver the results under default and improved tuning parameter settings. The default tuning parameter settings show the out-of-the-box performance. Based on these results we tried to improve the performance of each solver by changing the tuning parameters and – if necessary—the implementation of the log-likelihood function.

Table 2 gives an overview of the default and improved tuning parameter settings for each solver with the improved settings highlighted by a trailing asterisk. For ECOS we did not include an improved setting since the default setting did already work well. In the improved setting of `auglag` we did not only change the tuning parameter settings but also the implementation of the objective function and the gradient. For the implementation we replaced the logarithm in Eq. (3) with an extended version of the logarithm

$$\text{elog}(x) = \begin{cases} \log(x) & \text{if } x \geq 0 \\ -\infty & \text{if } x < 0. \end{cases}$$

This is necessary since in `auglag` the variable β may be outside the feasible region. Another interpretation of this modification is that we add the penalty term $-\infty$ whenever the constraint $X^0\beta \leq 0$ is violated. The code for the default version of the objective and the gradient and their improved counter parts are given in the appendix. For `constrOptim` and IPOPT this modification was not necessary since they operate in the interior of the feasible region. IPOPT depends on libraries which are not included

Table 2 Parameter settings for each solver

Solver	Max. number of iterations (outer/inner)	Method
auglag	50/100	BFGS
auglag*	10,000/10,000	BFGS
constrOptim	100/100	BFGS
constrOptim*	10,000/10,000	BFGS
ecos	100	
glm	25	
glm*	10,000	
glm2	25	
glm2*	10,000	
ipopt	3000	MA27
ipopt*	10,000	HSL_MA97
logbin	10,000	cem
logbin*	10,000	em

The improved settings are highlighted by a trailing asterisk

in the source code but have to be present at the installation of IPOPT. Particularly, IPOPT needs a linear solver to solve the augmented linear systems to obtain the search directions. Hereby the user can choose between eight different solvers. In the default setting we use the `MA27` solver which is the default of IPOPT and in the improved setting we use the `HSL_MA97` solver which is listed as the recommended solver on the HSL homepage.[2] Both solvers, `MA27` and `HSL_MA97`, are made available by the HSL mathematical software library under a proprietary license.

The solvers `auglag`, `constrOptim`, and IPOPT require the specification of starting values. For the solvers `glm` and `glm2` and those in package **logbin**, providing starting values is not strictly necessary. These solvers use their own routines for determining starting values, e.g., depending on the family and link and their initialization implementation. However, for `glm` and `glm2`, not providing starting values in some cases leads to the error message that the initialization did not lead to a valid set of coefficients. The ECOS solver does not require and does not allow the specification of starting values. All solvers except for `auglag` require that the starting values are in the interior of the parameter space.

There are different possibilities to determine starting values: de Andrade and de Leon Andrade (2018) suggest using the modified solution of a Poisson model given by $\hat{\beta}_0$ as starting value (i.e., $\hat{\beta}_0$ is modified such that the new $\hat{\beta}^*$ fulfills $X\hat{\beta}^* < 0$). Another possible approach is to first use a linear optimization solver to solve the problem consisting of the constraints $X\beta \leq 0$ and use the result as starting value. Finally, if an intercept is fitted, a simple possibility to choose a starting value is to use $(a, 0, \ldots, 0)$, with $a < 0$. Since we found no conclusive evidence that one of these initialization methods outperforms the others, we decided to use as starting value a special case of the simple approach given by $(-1, 0, \ldots, 0)$.

[2] http://www.hsl.rl.ac.uk/ipopt/.

4.3 Data set specific characteristics

4.3.1 A-priori assessment of uniqueness and finiteness of MLE

We use function `diagnose_lbrm` (shown in the appendix) to ensure that all the results reported for the different solvers are based on data sets for which the MLE is finite and unique. Two different situations can be distinguished for the different simulation scenarios: (1) the assumptions are met by all data sets generated or (2) the assumptions are met only by a subset of the data sets. In situation 1 the required number of data sets to repeat the application of the solvers is generated and it is reported that all data sets met the assumptions. In situation 2 more data sets than required for use with the solvers are generated to obtain reliable estimates for the proportion of data sets where the assumptions are met and only a subset of data sets which meet the assumptions of suitable size is used in combination with the different solvers.

4.3.2 A-posteriori assessment of the number of binding constraints

Because previous studies (e.g., de Andrade and de Leon Andrade 2018) indicated that binding constraints cause convergence problems, we compute the number of binding constraints after obtaining the MLE. For numerical reasons, we use the same approach as reported in de Andrade and de Leon Andrade (2018) to compute the number of binding constraints, i.e., counting the number of constraints where $\exp(X_{i*}\beta) \geq 0.9999$.

4.4 Simulations A

This simulation setting was introduced in Blizzard and Hosmer (2006) and used in Luo et al. (2014) and de Andrade and de Leon Andrade (2018). The simulation uses a simple univariate model

$$\log(P(Y = 1|x)) = \beta_0 + \beta_1 x \tag{21}$$

with a uniformly distributed covariate $x \sim U(-6, a)$, where the coefficients β_0, β_1 were chosen such that $P(Y = 1|x = -6) = 0.01$ and $P(Y = 1|x = 0) \in \{0.1, 0.3, 0.5, 0.7\}$. Furthermore, the values for a were chosen such that for each of the four pairs of coefficients there is a setting with low and high maximum probability $P(Y = 1|x = a)$. Based on that, eight different scenarios are considered. The corresponding coefficients are shown in Table 7.

4.4.1 Results

We checked all the generated data sets for separation and found that all data sets were overlapped and both X^0 and X^1 had full rank. This means that it is guaranteed that the solutions are finite and unique. Since binding constraints were identified to cause convergence problems in previous studies, we explicitly list the number of binding constraints for each scenario in Table 3.

Table 3 Simulation A: number of binding constraints (NBC)

NBC	Scenario							
	1	2	3	4	5	6	7	8
0	453	1000	844	1000	716	1000	477	1000
1	535	0	153	0	278	0	514	0
2	12	0	3	0	6	0	9	0

In this simulation experiment the main difference between the solvers is how often they converge with their default parameter settings compared to the improved tuning parameter settings. Table 9 shows the non-convergence rates in percent. We see that `glm` with the default values has the highest non-convergence rate but if we increase the maximum number of iterations the non-convergence rate decreases drastically. The results also indicate that in all cases where the solvers signaled success the solutions were admissible. With regard to BIAS, RMSE and CR we find that the results agree to those reported in previous studies. A summary of the results can be found in Table 10.

Figure 3 shows the ALLD for the converged results. We see that most solvers always give the global optimum if they signal success. Only in the default setting of `auglag` and the improved setting of `glm`/`glm2` the solvers signal success and return a suboptimal result. The improved solvers of `glm` and `glm2` may perform worse than the default versions because for the improved solvers instances are also included where the defaults did not converge and therefore were not considered in the evaluation of the ALLD. For `auglag` we see that changing the implementation details of the likelihood function influences how often the solver obtains the global solution.

4.5 Simulations B

In this simulation we implement the simulation scenarios proposed in Savu et al. (2010) and used in Luo et al. (2014) and de Andrade and de Leon Andrade (2018). These scenarios are designed to explore the behavior of the considered methods under misspecification. Scenarios 1–4 investigate the effect of link misspecification and scenarios 5–8 investigate the effect of linear predictor misspecification. In this simulation experiment we assume that the covariates $x_1 \sim$ Bernoulli(0.5), $x_2 \sim$ Multinomial(0.3, 0.3, 0.4), $x_3 \sim U(-1, 2)$ and the exposure status $E \sim$ Bernoulli(θ). The subject specific probability of exposure θ is given by

$$\theta = P(E = 1|x_1, x_2, x_3) = \text{logit}^{-1}(-1 + x_1 - x_{2(2)} + x_{2(3)} + x_3), \qquad (22)$$

with $x_{2(2)} = \mathbb{1}_{\{x_2=2\}}(x_2)$ and $x_{2(3)} = \mathbb{1}_{\{x_2=3\}}(x_2)$. The response variable is generated according to

$$P(Y = 1|E = 0, x) = g(a + h(x)),$$
$$P(Y = 1|E = 1, x) = 3\, g(a + h(x)),$$

Table 4 Simulation B: percentage of data sets that lead to infinite MLE's

Scenario							
1	2	3	4	5	6	7	8
0.55	0.21	0.07	58.89	0.00	0.00	0.00	0.07

Table 5 Simulation B: number of binding constraints (NBC)

NBC	Scenario							
	1	2	3	4	5	6	7	8
0	667	617	570	550	13	8	9	73
1	333	382	430	442	928	899	884	848
2	0	1	0	8	58	84	95	75
3	0	0	0	0	1	9	12	4

where a, g and h are scenario specific and can be found in Table 8. The choice $P(Y = 1|E = 1, x) = 3P(Y = 1|E = 0, x)$ ensures that for all scenarios the true adjusted relative risk of exposure RR_E is fixed to be the same and to be equal to 3.

4.5.1 Results

We find that in these simulation scenarios data sets are generated where some components of the MLE are infinite. In order to detect the infinite components we used the `diagnose_lbrm` function. Table 4 summarizes the results for the different configurations for 10,000 simulations and 500 observations. Especially for simulation scenario 4 the data generating process (DGP) generates data sets, where in more than half of the cases the MLE may have infinite components.

Since cases in which the MLE has infinite components are not well suited for the comparison of numerical optimization solvers, we restrict our simulations to cases where X^0 and X^1 have both full column rank.

Table 5 shows that these simulation scenarios yield more binding constraints than those in Simulation A.

Table 11 shows the non-convergence rates in percent for these simulations. The results are similar to those obtained for simulations A. Most of the solvers which have convergence issues for simulations B also had convergence issues for simulations A. Interestingly also ECOS failed to converge in 4 instances, looking more closely at the status messages we found that in all 4 instances ECOS failed with the error message "Numerical problems (unreliable search direction).". This issue occurs in rare cases, when the internally used linear solver cannot obtain the required precision. The BIAS, RMSE and CR can be found in Table 12. The values obtained are comparable to the results reported in previous studies. Figure 4 shows that the log-likelihood obtained by **logbin** is in many instances worse than the best log-likelihood obtained for an admissible solution. Furthermore, the improved versions of `glm` and `glm2` also return non-optimal values in some cases despite having signaled convergence.

Table 6 Simulation C: number of binding constraints (NBC)

NBC	Number of covariates			
	10	20	50	100
1	71	0	0	0
2	97	0	0	0
3–10	452	15	0	0
11–30	378	985	4	0
31–50	2	0	996	0
51–70	0	0	0	48
71–89	0	0	0	952

4.6 Simulations C

This simulation is designed to assess the speed and the reliability of the different solvers. To this end we consider a setting similar to Donoghoe and Marschner (2018), where we only change the sign of half of the β_i to allow generating simulation settings with up to 100 covariates. The data is generated by

$$\log(P(Y = 1|x)) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i \tag{23}$$

with $x_i \sim \text{Bernoulli}(0.5)$, $\beta_0 = \log(0.6)$ and

$$\beta_i = \begin{cases} -1 & \text{if } i \leq \frac{k}{2}, \\ 1 & \text{if } i > \frac{k}{2}. \end{cases}$$

Based on this we generated 1000 data sets, with 500 observations each and with $k = 10, 20, 50, 100$ covariates. For this simulation setting we do not report BIAS, RMSE and CR as these criteria are mainly useful for comparing different estimation methods but are less meaningful when comparing different solvers used for LBRM maximum likelihood estimation. Instead, we report the execution time in seconds and compare the ALLD obtained for the solvers based on varying tuning parameter settings (default versus improved) and different numbers of covariates.

4.6.1 Results

This scenarios were designed to be especially difficult, which we see on the high number of binding constrains shown in Table 6.

Figure 5 shows the ALLD distributions. For **logbin** we only report the improved results, since for more than 20 covariates **logbin** with the combinatorial EM (`"cem"`) method did not converge after several hours. The results show that `auglag`, ECOS and IPOPT always found the global optimum. Furthermore, we see that in this simulation setting in some cases `constrOptim` is not able to obtain the global optimum. A

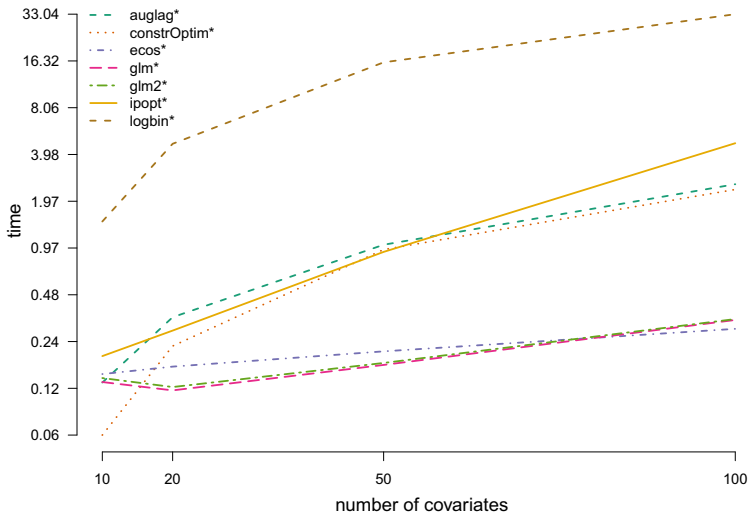

Fig. 2 Average time elapsed in seconds on a logarithmic scale

closer investigation shows that this is related to the fixed barrier term μ. In cases where the default value of μ is too low or too high for the given data set `constrOptim` does not converge to the global optimum. For `auglag` and `constrOptim` it is important to increase the maximum number of inner iterations from the default value `100` to `10000`. Furthermore, we find that the results of `auglag` strongly depend on the specific implementation of the likelihood function. Without modifying the implementation of the likelihood function, `auglag` fails to find the global optimum in more than half of the cases. For `constrOptim` and IPOPT this modification was not necessary since they operate in the interior of the feasible region.

4.6.2 Speed

Figure 2 shows the average elapsed times in seconds on the log scale for all solvers using the improved parameter settings. For all solvers the average elapsed time in seconds is smaller for 10 covariates than for 100 covariates and a general tendency of increase in elapsed time with increasing number of covariates is discernible.

Only the solvers with the improved parameter settings are compared to ensure that the comparison uses the solvers in a way when they are likely to return the global optimum. Note that Fig. 5 indicates that only ECOS, IPOPT and `auglag` with improved parameter settings were able to return the optimal solution in all cases. The timings for these three solvers are rather similar if there are only 10 covariates. However, ECOS clearly outperforms the other two solvers in computational efficiency if the number of covariates increases. In fact ECOS turns out to be among the quickest for 100 covariates while also reliably returning the global optimum. `glm` and `glm2` are also among the quickest for 100 covariates, but tend to return suboptimal solutions.

4.7 Weaknesses and strengths of the selected solvers

Overall two different forms of failure of a solver can be distinguished:

1. reporting non-convergence;
2. returning a non-optimal solution even when signaling convergence.

In the scenarios considered, reporting of non-convergence seems to be the smaller issue, since it most commonly occurs when the number of iterations was chosen too low. Thus, in many instances this problem can be resolved by the user by increasing the maximum number of iterations. If increasing the maximum number of iterations does not resolve the problem, the user can choose a different solver since they are aware that the obtained solution is not optimal. The issue that a solver returns a status code that signals convergence/success together with a non-optimal solution is much more concerning. This implies that the user is not aware of the solver returning a non-optimal solution to the problem.

4.7.1 `glm/glm2`

For the default setting of `glm` and `glm2`, we see that the non-convergence rates are high for most of the scenarios in the simulations A, B and C. Most of the cases where non-convergence occured can be explained by the fact that by default the maximum number of iterations is set to 25 which is typically too low when the solution is at the boundary of the parameter space. Therefore, in the improved setting, we increase the maximum number of iterations to 10,000. The results show that increasing the maximum number of iterations can resolve most of the non-convergence issues. Furthermore, the results show that in the improved setting `glm2` converges more often than `glm`. This can be accounted to the stricter step-halving criterion which addresses boundary and "repulsion" problems simultaneously. A "repulsion" problem refers to the case where the IWLS algorithm fails due to a repelling fixed point in the Fisher scoring iteration function. For more details, we refer to Marschner (2015). More concerning seems that Figs. 3, 4 and 5 indicate that `glm` and `glm2` in some instances signal convergence at a non-optimal likelihood.

4.7.2 logbin

The default and improved setting of **logbin** use different EM algorithms, with the other settings being the same. The non-convergence rates of the two algorithms are similar for simulations A and B. The `"cem"` algorithm is only applicable for a small number of covariates. Figures 4 and 5 show that in many instances, **logbin** returns a non-optimal solution despite of signaling convergence.

4.7.3 `auglag`

The performance of the `auglag` solver could be considerably improved by suitably modifying the implementation of the objective and the gradient. After changing the

objective such that it never returns `NaN` and changing the gradient function such that it always returns finite values, the `auglag` solver always converged and always returned the best log-likelihood found. However, as Figs. 3, 4 and 5 show, in the default setting, we could identify instances where `auglag` signaled convergence but did return a non-optimal solution. This indicates that it is possible that even in the improved setting it could happen that the solver signals convergence but returns a non-optimal solution.

4.7.4 `constrOptim`

The results show that `constrOptim` always signals convergence. However, as Fig. 5 shows even for the improved setting there exist instances where `constrOptim` returns a non-optimal solution while signaling convergence. Looking closer at the instances with non-optimal solutions, we find that in these instances the parameter μ was not chosen well enough. If we suitably modify the parameter μ, it is possible to obtain the optimal solution. Unfortunately, typically, we do not know how to choose μ to obtain the optimal solution.

4.7.5 IPOPT

Interestingly, the IPOPT solver fails to converge for a few instances in the default as well as in the improved setting. However, whenever the solver signaled success, it did return an optimal solution. Looking more closely into the failed convergence instances, we find that if we increase the maximum number of iterations up to 100,000, almost all instances converge.

4.7.6 ECOS

ECOS converges almost always. In four instances, it did not converge and reports as issue an unreliable search direction caused by numerical problems. In all instances where it signals success, it is able to retrieve the best solution found. This comes as no surprise since the self-dual embedding the ECOS solver uses allows ECOS to provide optimality certificates. Therefore, if ECOS returns the status code `0`, this does not only signal convergence but also optimality. This means in the LBRM context that, given that the MLE has only finite components, if ECOS signals optimality, the user can be certain that a global optimum was found. This is a key advantage of ECOS compared to solvers like `constrOptim` and `auglag` where the status code `0` only indicates successful completion of the optimization task.

5 Summary

In this paper, we gave an overview on necessary and sufficient conditions for the MLE of the LBRM to be finite and unique and indicated that it is important to consider these conditions when designing simulation studies—a fact which seems to have been neglected in previous simulation studies.

We gave an overview on possible optimization methods and solvers to be used for the LBRM and systematically compared them in three different simulation studies. The comparison involved using the default parameter settings to obtain the out-of-the-box behavior, but also improved parameter settings which implied that the solvers were more likely to return the global optimum.

The results of the simulation studies clearly imply that estimating the LBRM can profit from using conic optimization solvers. The main advantage of these solvers is their reliability. In the LBRM context, this means that, given that the conditions for finiteness and uniqueness of the MLE are fulfilled if the ECOS solver signals success the user can be sure that a global maximum was found. However, the results from the simulation study also indicate advantages with respect to the computational efficiency in particular when high-dimensional regression models are fitted.

Funding Open access funding provided by Vienna University of Economics and Business (WU).

Appendix A: Tables

Appendix A.1: Simulation settings

See Tables 7 and 8.

Appendix A.2: Simulation results

See Tables 9, 10, 11, 12 and 13, Figs. 3, 4 and 5.

Table 7 Simulation scenarios A

Scenario	β_0	β_1	a
1	−2.30259	0.38376	6
2	−2.30259	0.38376	4
3	−1.20397	0.56687	2
4	−1.20397	0.56687	1
5	−0.69315	0.65200	1
6	−0.69315	0.65200	0
7	−0.35667	0.70808	0.5
8	−0.35667	0.70808	−0.5

Table 8 Simulation scenarios B

Scenario	a	link (g^{-1})	h
1	−2.1	log	$-(x_1 + x_{2(2)} + x_{2(3)} + x_3)$
2	−1.9	cloglog	$-(x_1 + x_{2(2)} + x_{2(3)} + x_3)$
3	−1.7	logit	$-(x_1 + x_{2(2)} + x_{2(3)} + x_3)$
4	−1.48	probit	$-(x_1 + x_{2(2)} + x_{2(3)} + x_3)$
5	−1.1	log	$-\max(x_1 + x_{2(2)} + x_{2(3)} + x_3, 0)$
6	−0.9	cloglog	$-\max(x_1 + x_{2(2)} + x_{2(3)} + x_3, 0)$
7	−0.7	logit	$-\max(x_1 + x_{2(2)} + x_{2(3)} + x_3, 0)$
8	−0.48	probit	$-\max(x_1 + x_{2(2)} + x_{2(3)} + x_3, 0)$

Table 9 Results simulation A: NCR in percent

	Scenario							
	1	2	3	4	5	6	7	8
auglag	0.2	0.0	0.0	0.4	0.1	0.0	0.2	0.0
auglag*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
constrOptim	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
constrOptim*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ecos	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
glm	38.2	0.0	10.7	0.0	19.2	0.0	37.9	0.0
glm*	0.7	0.0	1.4	0.0	1.6	0.0	0.7	0.0
glm2	37.0	0.0	9.1	0.0	17.0	0.0	36.8	0.0
glm2*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ipopt	1.1	0.1	1.9	1.2	1.5	2.4	1.2	2.0
ipopt*	0.4	0.0	0.1	0.4	0.1	1.1	0.0	0.5
logbin	1.3	0.0	0.6	0.0	0.5	0.0	1.2	0.0
logbin*	1.3	0.0	0.6	0.0	0.5	0.0	1.2	0.0

Table 10 Results simulation A: BIAS, RMSE and CR of RR_1

		Scenario							
		1	2	3	4	5	6	7	8
RMSE	auglag	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	auglag*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	constrOptim	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	constrOptim*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	ecos	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	glm	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	glm*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	glm2	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	glm2*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	ipopt	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	ipopt*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	logbin	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10
	logbin*	0.03	0.06	0.05	0.08	0.05	0.09	0.05	0.10

Table 10 continued

		Scenario							
		1	2	3	4	5	6	7	8
BIAS	auglag	−0.39	0.68	0.37	1.07	0.24	1.33	−0.23	1.52
	auglag*	−0.18	0.68	0.37	1.04	0.24	1.33	−0.22	1.52
	constrOptim	−0.19	0.68	0.37	1.04	0.24	1.33	−0.23	1.52
	constrOptim*	−0.19	0.68	0.37	1.04	0.24	1.33	−0.23	1.52
	ecos	−0.18	0.68	0.37	1.04	0.24	1.33	−0.22	1.52
	glm	−0.66	0.68	−0.10	1.04	−0.48	1.33	−1.13	1.52
	glm*	−0.24	0.68	0.32	1.04	0.19	1.33	−0.40	1.52
	glm2	−0.62	0.68	−0.02	1.04	−0.37	1.33	−1.06	1.52
	glm2*	−0.23	0.68	0.38	1.04	0.23	1.33	−0.38	1.52
	ipopt	−0.19	0.65	0.28	0.85	0.19	1.00	−0.25	1.28
	ipopt*	−0.18	0.68	0.36	0.99	0.24	1.22	−0.22	1.52
	logbin	−0.18	0.68	0.38	1.04	0.23	1.33	−0.23	1.52
	logbin*	−0.18	0.68	0.38	1.04	0.23	1.33	−0.23	1.52
CR	auglag	95.00	95.50	96.00	96.10	95.80	95.70	96.10	95.70
	auglag*	96.10	95.50	96.00	96.10	95.80	95.70	96.10	95.70
	constrOptim	96.10	95.50	95.90	96.10	95.80	95.70	96.10	95.70
	constrOptim*	96.10	95.50	95.90	96.10	95.80	95.70	96.10	95.70
	ecos	96.10	95.50	96.00	96.10	95.80	95.70	96.10	95.70
	glm	96.60	95.50	97.00	96.10	96.90	95.70	97.10	95.70
	glm*	95.70	95.50	95.90	96.10	95.80	95.70	95.40	95.70
	glm2	96.50	95.50	96.90	96.10	96.70	95.70	97.00	95.70
	glm2*	95.60	95.50	95.90	96.10	95.80	95.70	95.40	95.70
	ipopt	96.10	95.60	96.10	96.30	95.80	96.30	96.10	96.10
	ipopt*	96.00	95.50	96.00	96.10	95.80	96.00	96.10	95.70
	logbin	96.00	95.50	95.90	96.10	95.80	95.70	96.00	95.70
	logbin*	96.00	95.50	95.90	96.10	95.80	95.70	96.00	95.70

Table 11 Results simulation B: NCR in percent

	Scenario							
	1	2	3	4	5	6	7	8
auglag	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
auglag*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
constrOptim	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
constrOptim*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ecos	0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0
glm	16.3	19.3	23.5	30.6	83.9	86.3	85.0	87.3
glm*	1.0	0.9	1.7	1.6	0.1	0.0	0.4	4.9
glm2	15.2	18.2	21.1	28.2	83.9	86.3	84.8	82.9
glm2*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ipopt	0.2	0.0	0.2	10.6	0.0	0.0	0.0	1.4
ipopt*	0.0	0.0	0.0	1.9	0.0	0.0	0.0	0.3
logbin	8.2	7.6	7.5	8.8	0.4	0.1	0.6	0.0
logbin*	7.1	6.0	6.4	11.5	0.6	0.3	0.7	0.1

Table 12 Results simulation B: BIAS, RMSE, and CR of RR_E

		Scenario							
		1	2	3	4	5	6	7	8
RMSE	auglag	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	auglag*	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	constrOptim	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	constrOptim*	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	ecos	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	glm	0.36	0.34	0.31	0.61	0.27	0.25	0.27	0.40
	glm*	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	glm2	0.36	0.34	0.31	0.61	0.27	0.25	0.27	0.41
	glm2*	0.36	0.33	0.31	0.60	0.28	0.26	0.26	0.39
	ipopt	0.36	0.33	0.31	0.59	0.28	0.26	0.26	0.39
	ipopt*	0.36	0.33	0.31	0.59	0.28	0.26	0.26	0.39
	logbin	0.39	0.34	0.31	0.65	0.34	0.32	0.31	0.50
	logbin*	0.34	0.32	0.30	0.55	0.34	0.32	0.31	0.50
BIAS	auglag	6.60	4.76	3.65	19.38	−13.93	−14.27	−14.21	−17.19
	auglag*	6.60	4.76	3.64	19.38	−13.86	−14.10	−14.20	−16.80
	constrOptim	6.59	4.74	3.63	19.37	−13.88	−14.12	−14.22	−16.81
	constrOptim*	6.59	4.74	3.63	19.37	−13.88	−14.12	−14.22	−16.81
	ecos	6.62	4.76	3.64	19.35	−13.86	−14.10	−14.20	−16.80
	glm	3.99	3.13	1.83	13.05	−12.37	−12.56	−14.66	−16.21

Table 12 continued

		Scenario							
		1	2	3	4	5	6	7	8
	glm*	6.48	4.82	3.50	18.96	−13.87	−14.14	−14.18	−16.35
	glm2	4.14	3.10	1.97	13.58	−12.37	−12.56	−14.66	−18.02
	glm2*	6.60	4.76	3.64	19.37	−13.88	−14.14	−14.22	−16.80
	ipopt	6.63	4.76	3.59	16.98	−13.86	−14.10	−14.20	−16.72
	ipopt*	6.60	4.76	3.64	18.92	−13.86	−14.10	−14.20	−16.77
	logbin	−1.26	−1.98	−2.63	−14.59	−23.07	−22.29	−21.62	−34.00
	logbin*	−0.42	−1.71	−2.67	−10.26	−23.09	−22.31	−21.63	−34.02
CR	auglag	94.90	94.70	93.50	93.50	76.70	76.20	75.30	62.50
	auglag*	94.90	94.70	93.50	93.50	76.40	75.60	75.10	62.90
	constrOptim	94.90	94.70	93.60	93.50	76.70	75.60	75.10	62.90
	constrOptim*	94.90	94.70	93.60	93.50	76.70	75.60	75.10	62.90
	ecos	94.90	94.70	93.50	93.50	76.40	75.60	75.10	62.90
	glm	95.60	95.00	93.90	95.20	79.50	78.10	70.70	63.80
	glm*	94.90	94.80	93.60	93.50	76.60	75.50	75.20	62.90
	glm2	95.50	95.00	93.80	95.30	79.50	78.10	71.10	61.40
	glm2*	94.90	94.70	93.50	93.50	76.50	75.50	75.10	62.90
	ipopt	94.90	94.70	93.50	93.50	76.40	75.60	75.10	62.90
	ipopt*	94.90	94.70	93.50	93.60	76.40	75.60	75.10	63.00
	logbin	94.20	96.10	96.60	97.80	88.90	86.10	86.10	86.60
	logbin*	97.20	97.00	96.80	98.30	88.90	86.10	86.10	86.60

Table 13 Results simulation C: NCR in percent

	Number of covariates			
	10	20	50	100
auglag	6.9	12.5	11.0	3.0
auglag*	0.0	0.0	0.0	0.0
constrOptim	0.0	0.0	0.0	0.0
constrOptim*	0.0	0.0	0.0	0.0
ecos	0.0	0.0	0.0	0.0
glm	99.9	99.9	99.5	97.9
glm*	1.6	3.3	4.4	3.2
glm2	99.9	99.9	99.5	97.9
glm2*	1.6	3.3	4.4	3.2
ipopt	0.1	0.0	0.0	0.0
ipopt*	0.5	0.0	0.0	0.0
logbin	1.5	–	–	–
logbin*	4.9	13.8	36.7	48.8

Due to timeout after 1 h the non-convergence rate is not available for logbin with more than 10 covariates

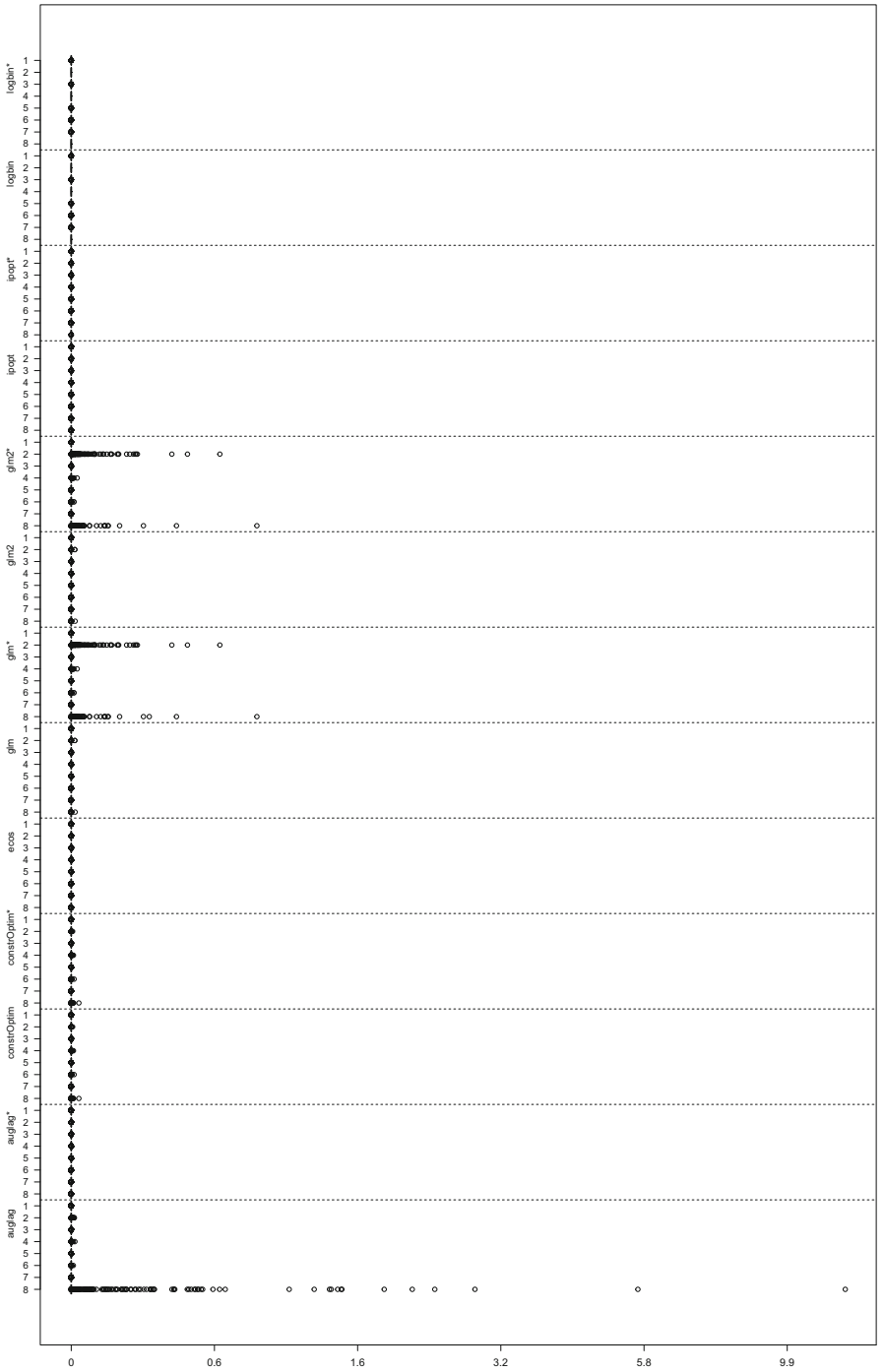

Fig. 3 The absolute difference to the best log-likelihood value for simulation A. Note here are only results considered where the solver signaled success

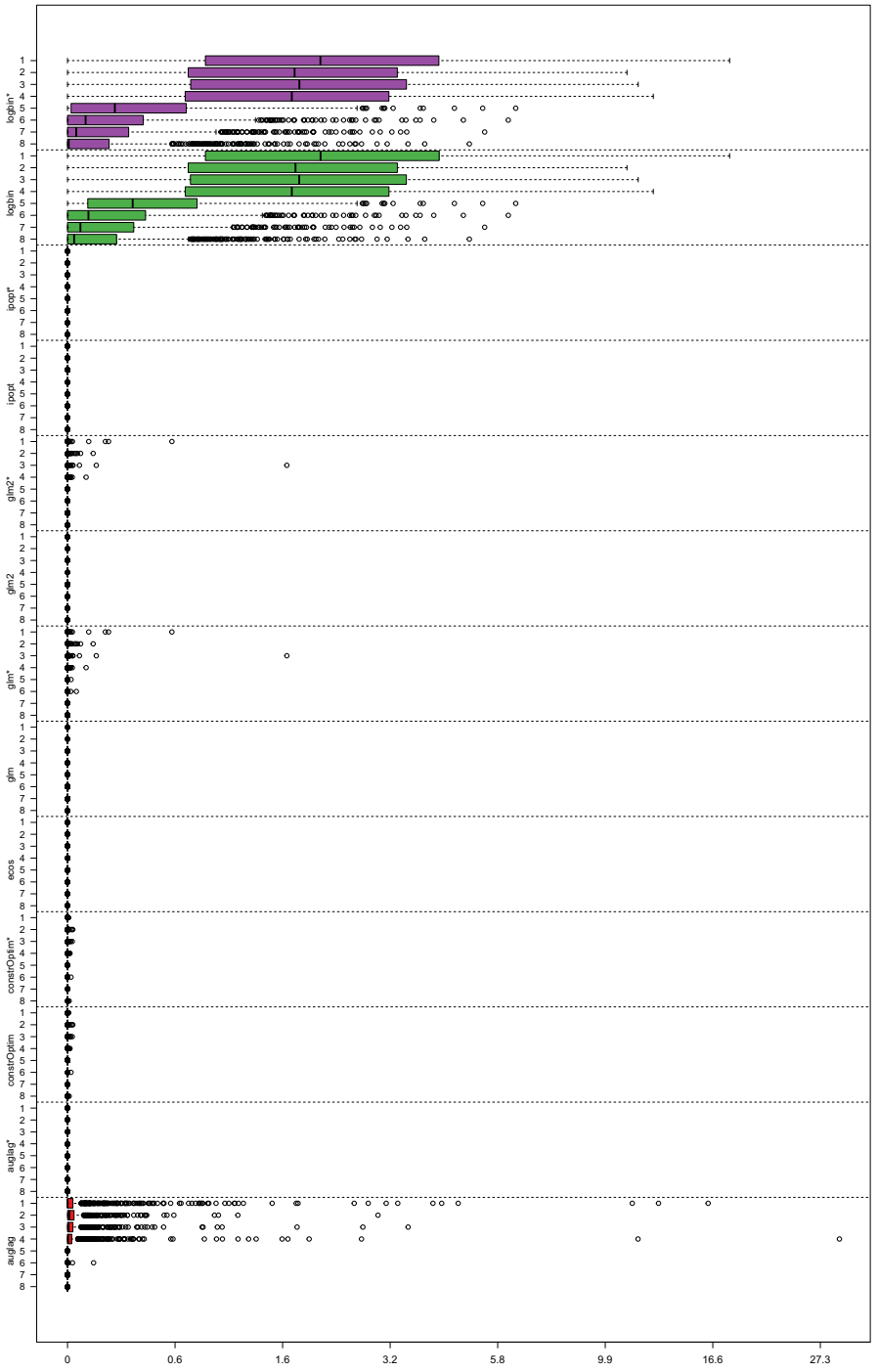

Fig. 4 The absolute difference to the best log-likelihood value for simulation B. Note here are only results considered where the solver signaled success

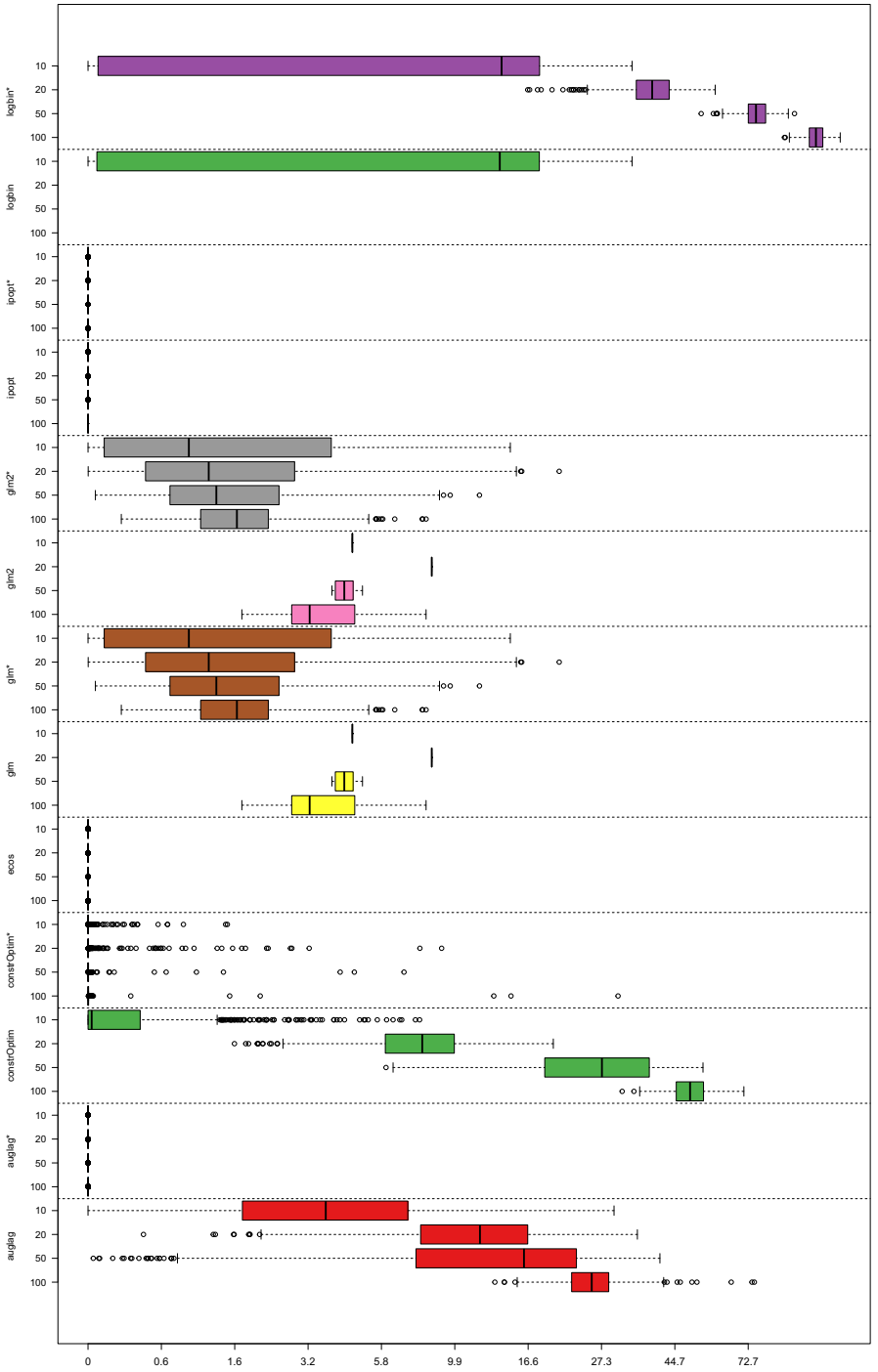

Fig. 5 The absolute difference to the best log-likelihood value for simulation C. Note here are only results considered where the solver signaled success

Appendix A.3: Functions

A.3.1 Default objective

```
neg_log_likelihood_default <- function(x, y, beta) {
  eta <- drop(tcrossprod(beta, x))
  -sum(dbinom(x = y, size = 1L, prob = exp(eta), log
    = TRUE))
}
```

A.3.2 Improved objective

```
elog <- function(x) {
  ans <- rep_len(-Inf, length(x))
  b <- x > 0
  ans[b] <- log(x[b])
  ans
}

neg_log_likelihood_improved <- function(x, y, beta) {
  eta <- drop(tcrossprod(beta, x))
  b <- (y == 1)
  -sum(eta[b]) - sum(elog(1 - exp(eta[!b])))
}
```

A.3.4 Default gradient

```
gradient_default <- function(beta) {
  mu <- exp(drop(tcrossprod(beta, x)))
  -drop(crossprod(x, (y - mu) / (1 - mu)))
}
```

A.3.5 Improved gradient

```
gradient_improved <- function(beta) {
  eta <- drop(tcrossprod(beta, x))
  mu <- pmin(exp(eta), 1 - .Machine$double.neg.eps)
  -drop(crossprod(x, (y - mu) / (1 - mu)))
}
```

A.3.6 Diagnose LBRM properties

```
library(ROI)
library(ROI.plugin.lpsolve)
```

```
diagnose_lbrm <- function(x, y) {
  properties <- list(rank = c(X = qr(x)$rank,
    X0 = qr(x[y==0,])$rank,
      X1 = qr(x[y==1,])$rank))

  if ( properties$rank["X"] < ncol(x) )
       stop("not enough data")

  if ( properties$rank["X1"] == ncol(x) ) {
    properties$infinite_components <- FALSE
  } else {
    op <- OP(-colSums(x))
    direction <- ifelse(y == 0, "<=", "==")
    constraints(op) <- L_constraint(x, direction,
     double(nrow(x)))
    bounds(op) <- V_bound(ld = -Inf, ud = Inf,
     nobj = ncol(x))
    maximum(op) <- TRUE
    ctrl <- list(pivoting = "firstindex", simplextype =
     c("primal", "primal"))
    s <- ROI_solve(op, solver = "lpsolve",
      control = ctrl)
    properties$infinite_components <- !isTRUE
    (solution(s, "status_code") == 0L)
  }
  return(properties)
}
```

References

Albert A, Anderson JA (1984) On the existence of maximum likelihood estimates in logistic regression models. Biometrika 71(1):1–10

Blizzard L, Hosmer W (2006) Parameter estimation and goodness-of-fit in log binomial regression. Biom J 48(1):5–22. https://doi.org/10.1002/bimj.200410165

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York

Calafiore GC, El Ghaoui L (2014) Optimization models. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9781107279667

Chares PR (2009) Cones and interior-point algorithms for structured convex optimization involving powers and exponentials. PhD thesis, Université Catholique de Louvain. http://dial.uclouvain.be/pr/boreal/en/object/boreal%3A28538/datastream/PDF_01/view

Davies HT, Crombie IK, Tavakoli M (1998) When can odds ratios mislead? BMJ (Clin Res Ed) 316(7136):989–991. https://doi.org/10.1136/bmj.316.7136.989

De Andrade BB, Carabin H (2011) On the estimation of relative risks via log binomial regression. Revista Brasileira de Biometria 29:1–15

de Andrade BB, de Leon Andrade JM (2018) Some results for maximum likelihood estimation of adjusted relative risks. Commun Stat Theory Methods 47(23):5750–5769. https://doi.org/10.1080/03610926.2017.1402045

De Leeuw J, Heiser W (1977) Convergence of correction matrix algorithms for multidimensional scaling. In: Lingoes J (ed) Geometric representations of relational data. Mathesis Press, Ann Arbor, chap 32, pp 735–753. http://deleeuwpdx.net/janspubs/1977/chapters/deleeuw_heiser_C_77.pdf
Domahidi A (2013) Methods and tools for embedded optimization and control. PhD thesis, ETH Zurich. https://doi.org/10.3929/ethz-a-010010483
Domahidi A, Chu E, Boyd S (2013) ECOS: an SOCP solver for embedded systems. In: European control conference (ECC), pp 3071–3076. https://web.stanford.edu/~boyd/papers/pdf/ecos_ecc.pdf
Donoghoe M, Marschner I (2018) logbin: an R package for relative risk regression using the log-binomial model. J Stat Softw 86(9):1–22 10.18637/jss.v086.i09
Fiacco AV, McCormick GP (1968) Nonlinear programming: sequential unconstrained minimization techniques. Wiley, New York
Holcomb WL, Chaiworapongsa T, Luke DA, Burgdorf KD (2001) An odd measure of risk: use and misuse of the odds ratio. Obstet Gynecol 98(4). https://journals.lww.com/greenjournal/Fulltext/2001/10000/An_Odd_Measure_of_Risk__Use_and_Misuse_of_the_Odds.28.aspx
Horn RA, Johnson CR (2012) Matrix analysis. Cambridge University Press, Cambridge
Hornik K, Meyer D, Schwendinger F, Theussl S (2020) ROI: R optimization infrastructure. R package version 1.0-0. https://CRAN.R-project.org/package=ROI
Hunter DR, Lange K (2004) A tutorial on MM algorithms. Am Stat 58(1):30–37
Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4(4):373–395. https://doi.org/10.1007/BF02579150
Kaufmann H (1988) On existence and uniqueness of maximum likelihood estimates in quantal and ordinal response models. Metrika 35(1):291–313. https://doi.org/10.1007/bf02613318
Konis K (2007) Linear programming algorithms for detecting separated data in binary logistic regression models. PhD thesis, University of Oxford. https://ora.ox.ac.uk/objects/uuid:8f9ee0d0-d78e-4101-9ab4-f9cbceed2a2a
Konis K, Fokianos K (2009) Safe density ratio modeling. Stat Probab Lett 79(18):1915–1920. https://doi.org/10.1016/j.spl.2009.05.020
Lange K (1994) An adaptive barrier method for convex programming. Methods Appl Anal 1(4):392–402
Lange K (2016) MM optimization algorithms. Soc Ind Appl Math 10(1137/1):9781611974409
Lumley T, Kronmal R, Ma S (2006) Relative risk regression in medical research: models, contrasts, estimators, and algorithms. Working Paper 293, UW Biostatistics Working Paper Series
Luo J, Zhang J, Sun H (2014) Estimation of relative risk using a log-binomial model with constraints. Comput Stat 29(5):981–1003. https://doi.org/10.1007/s00180-013-0476-8
Madsen K, Nielsen HB, Tingleff O (2004) Optimization with constraints, 2nd edn
Makhorin A (2011) GNU linear programming kit reference manual version 4.47. http://www.gnu.org/software/glpk
Marschner IC (2011) glm2: fitting generalized linear models with convergence problems. R J 3:12–15
Marschner IC (2015) Relative risk regression for binary outcomes: methods and recommendations. Aust N Z J Stat 57(4):437–462. https://doi.org/10.1111/anzs.12131
MOSEK ApS (2017) Introducing the MOSEK optimization suite. Version 8.1 (Revision 27). http://docs.mosek.com/8.1/intro/index.html
Nemirovski A (2006) Advances in convex optimization: conic programming. In: Proceedings of international congress of mathematicians, pp 413–444. https://doi.org/10.4171/022
Nesterov Y, Nemirovskii A (1994) Interior-point polynomial algorithms in convex programming. SIAM Studies in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia
Nocedal J, Wright SJ (2006) Numerical optimization. Springer, Berlin. https://doi.org/10.1007/978-0-387-40065-5
O'Donoghue B (2015) SCS–(splitting conic solver). https://github.com/cvxgrp/scs.git
O'Donoghue B, Chu E, Parikh N, Boyd S (2016) Conic optimization via operator splitting and homogeneous self-dual embedding. J Optim Theory Appl 1–27. https://doi.org/10.1007/s10957-016-0892-3
Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables, vol 30. Society for Industrial and Applied Mathematics
R Core Team (2020) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
Savu A, Liu Q, Yasui Y (2010) Estimation of relative risk and prevalence ratio. Stat Med 29(22):2269–2281. https://doi.org/10.1002/sim.3989

Serrano SA (2015) Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone. PhD thesis, Stanford University. https://web.stanford.edu/group/SOL/dissertations/ThesisAkleAdobe-augmented.pdf

Silvapulle MJ (1981) On the existence of maximum likelihood estimators for the binomial response models. J Roy Stat Soc Ser B (Methodol) 43(3):310–313

Theußl S, Schwendinger F, Hornik K (2020) ROI: an extensible R optimization infrastructure. J Stat Softw 94(15):1–64. https://doi.org/10.18637/jss.v094.i15

Varadhan R (2015) alabama: constrained nonlinear optimization. R package version 2015.3-1. https://CRAN.R-project.org/package=alabama

Wächter A (2009) Short tutorial: Getting started with Ipopt in 90 minutes. In: Naumann U, Schenk O, Simon HD, Toledo S (eds) Combinatorial scientific computing, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany, Dagstuhl, Germany, no. 09061 in Dagstuhl Seminar Proceedings. http://drops.dagstuhl.de/opus/volltexte/2009/2089

Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 106(1):25–57. https://doi.org/10.1007/s10107-004-0559-y

Williamson T, Eliasziw M, Fick GH (2013) Log-binomial models: exploring failed convergence. Emerg Themes Epidemiol 10:14. https://doi.org/10.1186/1742-7622-10-14