




## Holistic Generalized Linear Models

**Benjamin Schwendinger**   
Technische Universität  
Wien

**Florian Schwendinger**   
University of Klagenfurt

**Laura Vana**   
Technische Universität  
Wien

---

### Abstract

Holistic linear regression extends the classical best subset selection problem by adding additional constraints designed to improve the model quality. These constraints include sparsity-inducing constraints, sign-coherence constraints and linear constraints. The R package **holigm** provides functionality to model and fit holistic generalized linear models. By making use of state-of-the-art mixed-integer conic solvers, the package can reliably solve generalized linear models for Gaussian, binomial and Poisson responses with a multitude of holistic constraints. The high-level interface simplifies the constraint specification and can be used as a drop-in replacement for the `stats::glm()` function.

*Keywords:* algorithmic regression, best subset selection, conic programming, holistic constraints, optimization, R.

---

## 1. Introduction

Selecting a sensible model from the set of all possible models is an important but typically time-consuming task in the data analytic process. To simplify this process, [Bertsimas and King \(2015\)](#); [Bertsimas and Li \(2020\)](#) introduce the holistic linear model (HLM). The HLM is a constrained linear regression model in which the constraints aim to automate the model selection process. In particular, this can be achieved by utilizing quadratic mixed-integer optimization, where the integer constraints are used to place cardinality constraints on the linear regression model.

Cardinality constraints are used to introduce sparsity in the statistical model, a desirable property that leads to more interpretability, increase in computational efficiency and reduction in the variance of the estimates (at the cost of introducing some bias). Placing a cardinality constraint on the total number of variables allowed in the final model leads to the classical best subset selection problem ([Miller 2002](#)) which, given a response vector  $\mathbf{y} \in \mathbb{R}^n$ , a predictor matrix  $X \in \mathbb{R}^{n \times p+1}$  and a subset size  $k$  between 0 and  $\min\{n, p\}$ , finds the subset of at most

$k$  predictors  $\beta$  that produces the best fit in terms of squared error, solving the non-convex problem

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{y} - X\beta\|_2^2 \quad \text{subject to} \quad \sum_{j=1}^p \mathbb{I}_{\{\beta_j \neq 0\}} \leq k.$$

Several algorithms have been proposed to solve the best subset selection problem (among the most recent ones see e.g., [Hazimeh and Mazumder 2020](#); [Zhu, Wen, Zhu, Zhang, and Wang 2020](#)). Other approaches which aim to achieve global sparsity mostly rely on different types of penalties (such as LASSO-type penalties [Tibshirani 1996](#); [Sun and Zhang 2021](#)). Moreover, recent work considers similar constraints which be specified with the goal of introducing sparsity in the model at a group level, i.e., ensuring that certain sets of predictors are jointly included (excluded) in (from) the model. [Hu, Li, Meng, Qin, and Yang \(2017\)](#) introduce sparsity in a group structure via the  $\ell_{p,q}$  norm by using group restricted eigenvalue condition, while ([Zhang, Zhu, Zhu, and Wang 2022b](#)) propose a group-splicing algorithm that iteratively detects the relevant groups and excludes the irrelevant ones. Further cardinality constraints placed on user-defined groups of predictors can be used to e.g., limit the pairwise multicollinearity, select the best (non-linear) transformation for the predictors or include expert knowledge in the model estimation (such as forcing specific predictors to stay in the model).

In addition to cardinality constraints, upper or lower bounds as well as linear constraints on the coefficients are also relevant for improving interpretability along with introducing sparsity (e.g., see [Slawski and Hein 2013](#), who introduce non-negativity constraints in linear regression and show performance similar to other sparsity inducing methods).

In this paper, we introduce the **holiglm** package, an R package for formulating and fitting holistic generalized linear models (HGLMs). The package supports holistic constraints such as mixed-integer-type constraints related to sparsity (e.g., limits on the number of predictors to be included in the model, group sparsity constraints), linear constraints and constraints related to multicollinearity. The contribution of the paper is three-fold. First, to the best of our knowledge, we are the first to suggest the use of conic optimization to extend the results presented for linear regression by [Bertsimas and King \(2015\)](#); [Bertsimas and Li \(2020\)](#) to the class of generalized linear models (GLMs) and to formulate the most common ones as conic optimization problems. Secondly, we survey the literature related to holistic linear regression and, more generally, to constrained regression and provide an extensive survey of what should be considered as holistic constraints. Finally, we provide a ready-to-use implementation of holistic GLMs in the package **holiglm**. We exemplify the use of the package for a variety of statistical problems with the hope that this will encourage the statistical community in further exploiting the recent advances in mixed-integer conic optimization.

The **holiglm** package provides a flexible infrastructure for automatically translating constrained generalized linear models into conic optimization problems. The optimization problems are solved by utilizing the R optimization infrastructure package **ROI** ([Theußl, Schwendinger, and Hornik 2020](#)). Additionally, a high-level interface, which can be used as a drop-in replacement for the `stats::glm()` function, is provided. Using **ROI** makes it possible for the user to choose between a wide range of commercial and open source optimization solvers. With recent advancements in conic optimization, it is now possible to routinely solve conic problems with a large number of variables and/or constraints to proven optimality. Using conic optimization instead of iteratively reweighted least squares (IRLS)

has the advantages that no starting values are needed, the results are more reliable and the solvers are designed to handle different types of constraints. These advantages come at the cost of a longer runtime; however, as shown by Schwendinger, Grün, and Hornik (2021) for some GLMs the speed of the conic formulation is similar to the IRLS implementation. Nevertheless, problems with (very) large  $n$  and/or (very) large  $p$  can currently pose a marked computational burden on the software. It should be noted that the computational burden depends on both the conic formulation itself (in Appendix A one can see the dimension of the underlying optimization problems for different family and link functions) as well as the computational ability of the solvers.

At the time of writing, there exists no ready-to-use package or library for fitting HGLMs or HLMs. However, for R (R Core Team 2023) several packages allow the estimation of (generalized) linear models under linear or sparsity constraints. Package **abess** (Zhu *et al.* 2022) can fit generalized linear models with canonical link functions while enforcing global and group sparsity constraints. The **CMLS** (Helwig 2018) package can fit linear regression models under linear constraints by translating the problem into a linear constrained quadratic optimization problem, which is solved by **quadprog** (Turlach, Weingessel, and Moler 2019). Package **colf** (Boutaris 2017) can fit linear regression models with lower and upper bounds on the coefficients. The **glm** (Chaudhuri, Handcock, and Rendall 2006) package can fit generalized linear models while imposing linear constraints on the parameters. Similarly, package **restriktor** (Vanbrabant 2022) provides functionalities for estimation, testing and evaluation of linear equality and inequality constraints about parameters and effects for generalized linear models, where the constraints can be specified by a text-based description. Finally, **ConsReg** (Sallés 2020) provides a similar functionality to **restriktor**. Package **holiglm** (Schwendinger, Schwendinger, and Vana 2024) is available from the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/package=holiglm>.

This paper is structured as follows: Section 2 introduces the GLM class and presents the representation of specific family-link combinations as conic optimization models. In Section 3, we present a list of holistic constraints implemented in the package. Section 4 introduces package **holiglm** for R. In Section 5 we compare our package regarding runtimes with existing packages for fitting constrained GLMs. Section 6 shows the usage of the package in two applications: fairness constraints in logistic regression and model selection in log-binomial regression. Section 7 concludes.

## 2. Generalized linear models

A generalized linear model is a model for a response variable whose conditional distribution belongs to a one-dimensional exponential family. A GLM consists of a linear predictor

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \dots + \beta_p x_{pi} = \mathbf{x}_i^\top \boldsymbol{\beta}$$

and two functions, namely a twice continuously differentiable and invertible link function  $g$  that describes how the mean,  $\mathbf{E}(\mathbf{y}) = \boldsymbol{\mu}$ , depends on the linear predictor (one has  $g(\boldsymbol{\mu}) = \boldsymbol{\eta}$ ), and a variance function  $V$  that describes how the variance depends on the mean  $\text{VAR}(\mathbf{y}) = \phi V(\boldsymbol{\mu})$  (where the dispersion parameter  $\phi > 0$  is a constant).

The most common distributions for the response  $\mathbf{y}$  that can be accommodated in the GLM setting include the normal, binomial or Poisson. The density of members of the exponential

| Family-link           | Functions<br>$\lambda_i(\boldsymbol{\beta}), b(\lambda_i(\boldsymbol{\beta}))$  | Objective  | Conic constraint   |
|-----------------------|---|--|--|
| <b>Gaussian</b>       |   |  |  |
| Identity <sup>a</sup> | $\lambda_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta},$<br>$b(\lambda_i(\boldsymbol{\beta})) = (\mathbf{x}_i^\top \boldsymbol{\beta})^2 / 2$  | $-\zeta$   | $(\zeta + 1, \zeta - 1, 2(y_1 - \mathbf{x}_1^\top \boldsymbol{\beta}), \dots, 2(y_n - \mathbf{x}_n^\top \boldsymbol{\beta})) \in \mathcal{K}_{\text{soc}}^{n+2}$   |
| <b>Binomial</b>       |   |  |  |
| Logit <sup>b</sup> &  | $\lambda_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta}$  | $\sum_{i=1}^n y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \delta_i$           | $(\delta_i, 1, \gamma_i + 1) \in \mathcal{K}_{\text{exp}}$   |
| Probit                | $b(\lambda_i(\boldsymbol{\beta})) = \log(1 + \exp(\mathbf{x}_i^\top \boldsymbol{\beta}))$   |  | $(\mathbf{x}_i^\top \boldsymbol{\beta}, 1, \gamma_i) \in \mathcal{K}_{\text{exp}}$   |
| Log <sup>b</sup>      | $\lambda_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta} - \log(1 - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}))$<br>$b(\lambda_i(\boldsymbol{\beta})) = -\log(1 - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}))$ | $\sum_{i=1}^n y_i \mathbf{x}_i^\top \boldsymbol{\beta} + (1 - y_i) \delta_i$ | $\mathbf{x}_i^\top \boldsymbol{\beta} \leq 0$<br>$(\delta_i, 1, 1 - \gamma_i) \in \mathcal{K}_{\text{exp}}$<br>$(\mathbf{x}_i^\top \boldsymbol{\beta}, 1, \gamma_i) \in \mathcal{K}_{\text{exp}}$                                      |
| <b>Poisson</b>        |   |  |  |
| Log <sup>b</sup>      | $\lambda_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta}$<br>$b(\lambda_i(\boldsymbol{\beta})) = \exp(\mathbf{x}_i^\top \boldsymbol{\beta})$   | $\sum_{i=1}^n y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \delta_i$           | $(\mathbf{x}_i^\top \boldsymbol{\beta}, 1, \delta_i) \in \mathcal{K}_{\text{exp}}$   |
| Identity <sup>b</sup> | $\lambda_i(\boldsymbol{\beta}) = \log(\mathbf{x}_i^\top \boldsymbol{\beta})$<br>$b(\lambda_i(\boldsymbol{\beta})) = \mathbf{x}_i^\top \boldsymbol{\beta}$   | $\sum_{i=1}^n y_i \delta_i - \mathbf{x}_i^\top \boldsymbol{\beta}$           | $(\delta_i, 1, \mathbf{x}_i^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{exp}}$<br>$\mathbf{x}_i^\top \boldsymbol{\beta} \geq 0$  |
| Sq.root <sup>ab</sup> | $\lambda_i(\boldsymbol{\beta}) = 2 \log(\mathbf{x}_i^\top \boldsymbol{\beta})$<br>$b(\lambda_i(\boldsymbol{\beta})) = (\mathbf{x}_i^\top \boldsymbol{\beta})^2$   | $\sum_{i=1}^n 2y_i \delta_i - \zeta$   | $(\delta_i, 1, \mathbf{x}_i^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{exp}}$<br>$(\zeta + 1, \zeta - 1, 2\mathbf{x}_1^\top \boldsymbol{\beta}, \dots, 2\mathbf{x}_n^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{soc}}^{n+2}$ |

Table 1: GLMs as exponential families with conic reformulations. The objective functions to be maximized are proportional to  $\propto \sum_{i=1}^n y_i \lambda_i(\boldsymbol{\beta}) - b(\lambda_i(\boldsymbol{\beta}))$ . The conic constraints with subscript  $i$  should hold for all  $i = 1, \dots, n$ . Superscript <sup>a</sup> marks family-link combinations with an objective modeled by second-order cones. Superscript <sup>b</sup> marks family-link combinations with an exponential cone representation. Note that for the probit link an approximation is used by scaling the coefficients obtained from the logit link by  $\sqrt{\pi/8}$  which is equivalent to using a simple approximation (Page 1977) for the standard normal distribution.

family can be written as:

$$f(\mathbf{y}; \boldsymbol{\lambda}, \phi) = \exp \left\{ \frac{\mathbf{y}\boldsymbol{\lambda} - b(\boldsymbol{\lambda})}{\phi} + c(\mathbf{y}, \phi) \right\},$$

where  $\boldsymbol{\lambda}$  is called the canonical or natural parameter,  $b(\cdot)$  and  $c(\cdot)$  are known real-valued measurable functions that vary from one exponential family to another.

It can be shown that  $\mathbf{E}(\mathbf{y}) = b'(\boldsymbol{\lambda})$  and  $\text{VAR}(\mathbf{y}) = \phi b''(\boldsymbol{\lambda})$  holds. Note that  $g(b'(\boldsymbol{\lambda})) = \boldsymbol{\eta}$ . We can denote  $h = (b')^{-1} \circ g^{-1}$  such that, when rewriting  $\boldsymbol{\lambda}$  as a function of  $\boldsymbol{\beta}$  we have  $\lambda(\boldsymbol{\beta}) = h(\boldsymbol{\eta})$ . In case  $h$  is the identity, the link  $g$  is called canonical, i.e.,  $b'(\cdot)$  is the inverse of the canonical link function.

The parameters  $\boldsymbol{\beta}$  can be estimated by maximizing the likelihood function corresponding to the different types of response. Using the distribution of the exponential family, the log-likelihood for a sample  $\mathbf{y} = (y_1, \dots, y_n)^\top$  is given by:

$$\log \mathcal{L}(\boldsymbol{\beta}; \mathbf{y}) = \sum_{i=1}^n \frac{y_i \lambda_i(\boldsymbol{\beta}) - b(\lambda_i(\boldsymbol{\beta}))}{\phi_i} + c(y_i, \phi_i). \quad (1)$$

where  $\phi_i = \phi/a_i$  for  $a_i$  are user-defined observation weights.

In this paper, we leverage the fact that the maximization of the objective function in Equation 1 can be reformulated as a conic optimization problem, for most common family-link combinations. A conic optimization problem is designed to model convex problems by optimizing a linear objective function over the intersection of an affine hyperplane and a nonempty closed convex cone. For GLMs, the types of cones relevant for maximum likelihood estimation include the exponential cone  $\mathcal{K}_{\text{exp}}$ , which can be used to model a variety of objectives and constraints involving exponentials and logarithms, and the second-order cone  $\mathcal{K}_{\text{soc}}$ , which can be used to model problems involving – directly or indirectly – quadratic terms. In **holigm** we provide functionality for estimating GLMs with Gaussian, binomial and Poisson families with various link functions. Table 1 contains information on the family-link combinations implemented in package **holigm**. The second column contains, for each  $i$ , the  $\lambda_i(\cdot)$  and  $b_i(\cdot)$  as functions of the vector of coefficients  $\beta$ . The third column presents the objective function obtained by reformulating the maximization of the log-likelihood as a conic program, while the fourth column contains the corresponding conic constraints. Note that all reformulations necessitate the specification of auxiliary variables  $\zeta \in \mathbb{R}$  and/or  $\delta \in \mathbb{R}^n$ , which will be optimized together with the vector of regression coefficients. More details on the reformulation of the log-likelihood maximization problem for the presented family-link combinations can be found in Appendix A. Furthermore, we provide a package vignette which discusses how the user can decide whether a problem is solvable via conic optimization, i.e., whether the objective (and desired constraints) can be represented using convex cones. Note that the package currently does not allow for inverse Gaussian, gamma or negative binomial families, as at the time of writing no appropriate conic representation could be found for these families. More information about conic optimization in general can be found in Boyd and Vandenberghe (2004) and MOSEK ApS (2022a), more information about conic optimization in R can be found in Theußl *et al.* (2020).

### 3. Holistic constraints

Bertsimas and King (2015) and Bertsimas and Li (2020) introduced a set of constraints designed to improve the quality of linear regression models. These constraints are crafted based on a survey of modeling recommendations from statistical textbooks and articles. In this section we survey different constraints suggested by the literature and provide an extensive overview which includes but is not limited to constraints in Bertsimas and King (2015) and Bertsimas and Li (2020). Moreover, we provide their formulations as optimization constraints. Most of the constraints introduced in **holigm** are cardinality constraints. For modeling cardinality constraints we need  $p$  binary variables:

$$z_j \in \{0, 1\}, \quad j = 1, \dots, p. \quad (2)$$

These binary variables are only added to the model if needed. Here, the binary variable  $z_j$  represents the selection of variable  $X_j$ , hence,  $z_j = 0$  implies  $\beta_j = 0$ . Problems which involve such binary variables are called mixed-integer optimization problems. This type of cardinality constraints can be modeled with a so-called big- $M$  constraint.

$$-Mz_j \leq \beta_j \leq Mz_j, \quad j = 1, \dots, p, \quad (3)$$

where  $M$  is a positive constant. For the big- $M$  constraint it is important to choose a good constant  $M$ . A good  $M$  can be characterized by two properties:

- It is chosen big enough, that it does not impact the magnitude of the parameter  $\beta_j$ .
- It is chosen as small as possible to improve the speed of the underlying optimization solver and ensure stability.

### 3.1. Global sparsity

The global sparsity constraint can be used to model the classical best subset selection problem (Miller 2002). The best subset selection problem is a combinatorial optimization problem designed to select a predefined number of  $k_{\max}$  predictors out of all available predictors. This can be accomplished by maximizing the likelihood function while restricting the number of predictors via integer constraints:

$$\sum_{j=1}^p z_j \leq k_{\max}, \quad \text{where } k_{\max} \leq p, \quad k_{\max} \in \mathbb{N}.$$

### 3.2. Group sparsity

Contrary to global sparsity constraints, group sparsity constraints do not restrain all predictors but only specific local groups of predictors. Group sparsity constraints can again be subdivided into excluding and including constraints:

- Excluding constraints enforce a cardinality constraint on a subset of predictors.
- Including constraints enforce that either all predictors of a group are selected or none.

Bertsimas and King (2015) suggest to use excluding constraints for modeling pairwise multicollinearity and selecting the best (non-)linear transformation. Including constraints can be used for modeling categorical variables via one-hot encoding or splines via spline basis functions. In the following we present the formulation of the excluding group sparsity constraints for restricting pairwise multicollinearity and for selecting the best variable transformation.

#### *Limited pairwise multicollinearity*

A desired property for a good model is that the predictors exhibit no multicollinearity. Collinear predictors can cause numerical problems during the estimation, less interpretable coefficients and misleading standard errors.

The group sparsity constraint can be used to restrict pairwise multicollinearity by excluding pairs of predictors that exhibit strong pairwise correlation.

Using the previously introduced variables  $z$ , the pairwise multicollinearity can be formulated as,

$$z_j + z_k \leq 1 \quad \forall (j, k) \in \mathcal{PC},$$

here  $\mathcal{PC} = \{(j, k) \mid \rho_{\max} < |\rho(j, k)|\}$  denotes the set of pairs of highly correlated predictors. The general rule of thumb is excluding pairs of predictors where the absolute value of the

pairwise correlation exceeds the threshold  $\rho_{\max} = 0.7$ , but other – less as well as more restrictive – choices are also advocated for in the literature (see [Dormann \*et al.\* 2013](#), for a review).

### *Non-linear transformations*

In cases where the true relationship between the response and a covariate is non-linear, using an appropriate transformation can improve model quality. Commonly, non-linear transformations like  $\log(x)$ ,  $\sqrt{x}$  or  $x^2$  are used. When using non-linear transformations, it is often desirable to only include either the non-transformed variable or only one of the transformed variables. A group sparsity constraint of the form

$$\sum_{j \in \mathcal{NL}} z_j \leq 1,$$

can be used to select at most one of the transformations. Here  $\mathcal{NL}$  denotes the set of indices of applied transformations, including the identity mapping, on a certain covariate.

### *Including group constraints*

In cases where all predictors within a specified group  $\mathcal{G}$  should be included in the model (as in the case of dummy-encoded categorical variables), the including group sparsity constraint (also referred here as in-out constraint) is translated to equality of all corresponding binary variables:

$$z_j = z_k \quad \forall (j, k) \in \mathcal{G}.$$

## **3.3. Modeler expertise**

[Bertsimas and King \(2015\)](#) point out that, in some situations, the modeler knows the true importance of a particular covariate, be it some business requirement or other more profound expert knowledge. In these cases, the modeler can choose to force the inclusion of a particular covariate. This might be necessary when the automated covariate selection process decides to discard the covariate due to other global or group sparsity constraints. To achieve this inclusion, the following constraint can be used:

$$z_j = 1.$$

## **3.4. Bounded domains for predictors**

In certain applications, it is of interest to set constraints (bounds) on the value of particular regression coefficients. For example, [McDonald and Diamond \(1990\)](#) consider the problem of finding maximum likelihood estimates of a generalized linear model when some or all regression parameters are constrained to be non-negative and motivate their approach with applications in cancer death prediction and demography. Similarly, [Slawski and Hein \(2013\)](#) propose the use of non-negative least squares in high-dimensional linear models. They show that this can have similar effects to explicit regularization like LASSO ([Tibshirani 1996](#)) or ridge regression ([Hoerl and Kennard 1970](#)) regarding predictive power and sparsity when mild conditions on the design matrix are met. Moreover, there exist problems where the sign of the

effect of a covariate is known beforehand. This allows the restriction of coefficient domains to only non-negative or only non-positive values (Carrizosa, Olivares-Nadal, and Ramírez-Cobo 2020). In our conic model approach, for a coefficient  $\beta_j$ , we can add bounds (sometimes also called box constraints)  $l_j$  and  $u_j$  by the following constraint:

$$l_j \leq \beta_j \leq u_j.$$

### 3.5. Linear constraints

As Lawson and Hanson (1995) point out, it might also be desirable to enforce linear constraints of the type  $L\boldsymbol{\beta} \leq \mathbf{c}$  for certain predictors (typical constraints include  $\beta_j + \beta_k \leq \beta_l$ ,  $\beta_j \leq \beta_k$  or  $\beta_j = \beta_k = \beta_l$ ). Such constraints might arise in applied mathematics, physics and economics and usually convey additional information about a problem. Adding additional convex constraints to a problem will only tighten the set of solutions. One still has to be careful not to add constraints that make the underlying optimization problem infeasible. If any infeasibility arises, the underlying optimization solver will detect it.

Note that such linear constraints can also be imposed on the binary variables. For example, constraints on the binary variable can be used to model if-then types of relations. As an example, consider a constraint of the type “if the  $j$ -th predictor is included in the model then the  $k$ -th predictor should also be included in the model”. This would translate to the constraint  $z_j \leq z_k$ . Note also that the global and group sparsity constraints introduced above are special cases of linear constraints on the binary variables.

### 3.6. Sign coherence constraint

Sign coherence can be a desirable model property which improves interpretability, especially in the presence of highly correlated predictors. Carrizosa *et al.* (2020) suggest using a sign coherence constraint instead of the group sparsity constraint to restrict pairwise multicollinearity. They argue that enforcing sign coherence (forcing strongly positive correlated variables to have the same sign and strongly negative correlated variables to have opposite signs) is less restrictive than using a group sparsity constraint, since it allows strongly correlated variables to be jointly in the model without lowering interpretability due to inconsistent signs. Without this restriction, inconsistent signs of strongly correlated variables are often caused by compensating coefficients (Hahs-Vaughn 2016). Given the set  $\mathcal{PC}$  of pairs of indices of highly correlated predictors, we can enforce equal signs of their respective coefficients by adding the constraints

$$-M(1 - u_{jk}) \leq \beta_j, \quad \text{sign}(\rho_{jk})\beta_k \leq Mu_{jk}, \quad \forall (j, k) \in \mathcal{PC},$$

where again  $M$  is a large enough constant and  $u_{jk}$  is a newly introduced binary variable. One can see that for  $u_{jk} = 0$  and  $\rho_{jk} > 0$  we have  $-M \leq \beta_j, \beta_k \leq 0$ , while for  $u_{jk} = 1$ , we have  $0 \leq \beta_j, \beta_k \leq M$  for an arbitrary pair  $(j, k)$  in  $\mathcal{PC}$ .

## 4. The holiglm package

Package **holiglm** allows to reliably and conveniently fit generalized linear models under constraints. We aimed to allow for as many family-link combinations and constraints as possible without reducing the reliability of the solution. To accomplish these goals, the package uses



| Link/Family | Gaussian | Binomial  | Poisson   |
|-------------|----------|-----------|-----------|
| Identity    | Q   SOC  |           | LIN & EXP |
| Log         |          | LIN & EXP | EXP       |
| Logit       |          | EXP       |           |
| Probit      |          | EXP       |           |
| Square root |          |           | SOC & EXP |

Table 2: Overview of the cones needed to express specific GLMs. Here, Q | SOC indicates that either a quadratic solver or a solver supporting the second-order cone can be used. SOC & EXP indicates that the second-order cone and exponential cone are used to model the Poisson model with power link. Note that for the probit link an approximation is used by scaling the coefficients obtained from the logit link by  $\sqrt{\pi/8}$  (Page 1977).

| Solver | License     | Package                  | Quadratic | Objective | LIN | SOC | EXP |
|--------|-------------|--------------------------|-----------|-----------|-----|-----|-----|
| CPLEX  | Proprietary | <b>ROI.plugin.cplex</b>  |           | ✓         |     |     |     |
| ECOS   | GPL-3       | <b>ROI.plugin.ecos</b>   |           |           | ✓   | ✓   | ✓   |
| GUROBI | Proprietary | <b>ROI.plugin.gurobi</b> |           | ✓         |     |     |     |
| MOSEK  | Proprietary | <b>ROI.plugin.mosek</b>  |           | ✓         | ✓   | ✓   | ✓   |
|        | Mixed       | <b>ROI.plugin.neos</b>   |           | ✓         |     |     |     |

Table 3: Overview of the solvers implemented as plugins in **ROI** that can handle mixed-integer constraints in combination with either a quadratic objective with linear constraints or a linear objective with linear, second-order and exponential conic constraints.

state-of-the-art mixed-integer (conic) solvers. Using conic optimization, we can reliably solve convex non-linear mixed-integer problems, given that there exists a combination of cones that can express the non-linear problem at hand. Luckily, the (log-)likelihood of most GLMs can be expressed by combinations of the linear (non-negative) cone, the second-order cone and the exponential cone. Table 2 gives an overview on the cones used to express specific GLMs. Currently, the solvers **ECOS** (Domahidi, Chu, and Boyd 2013), **MOSEK** (**MOSEK ApS 2022b**) **CPLEX** (IBM ILOG 2022), **GUROBI** (Gurobi Optimization LLC 2022) and the **NEOS** (Czyzyk, Mesnier, and Moré 1998) Server via the **ROI.plugin.neos** (Hochreiter and Schwendinger 2020) can be used if the model contains mixed-integer constraints. Table 3 provides additional information on the solver licenses and on the different types of cones supported by these solvers. All the solvers listed in Table 3 internally aim to prove the optimality of the solution by checking criteria based on the Karush-Kuhn-Tucker optimality conditions. Consequently, if these solvers signal optimality, the user can be certain that the maximum likelihood estimate (MLE) was obtained, except for the special case where the MLE does not exist. In theory, these solvers should be able to provide a certificate of unboundedness if the MLE does not exist; however, practically this is often not the case for the exponential cone. Fortunately, package **detectseparation** (Kosmidis, Schumacher, and Schwendinger 2022) can be used to verify the existence of the MLE before the estimation. More information on conic optimization in general can be found in Boyd and Vandenberghe (2004).

In R, the packages **ROI** and **CVXR** (Fu, Narasimhan, and Boyd 2020) provide access to multiple conic solvers. The **CVXR** package provides a domain-specific language (DSL) which allows to formulate the optimization problems close to their mathematical formulation. To accomplish this, it first translates the problem into an abstract syntax tree and uses disciplined

convex programming (DCP, Grant, Boyd, and Ye 2006) to verify that the problems are indeed convex. After the convexity is verified, the problem is transformed into its matrix representation and dispatched to the selected solver. Here it is important to note that the DCP rules are only sufficient for convexity, meaning that even if the DCP rules can not verify convexity, the problem can still be convex. Based on these properties, from our perspective the **CVXR** package is especially well-suited for users unfamiliar with conic optimization or fast prototyping. The **ROI** package, on the other hand, offers a unified solver access to many solvers, with a simple modeling language designed for users already familiar with R. For the **holiglm** package, we choose to rely on the infrastructure of package **ROI** since we already know the convexity properties of the likelihood functions and we want to be able to control the transformation of the likelihood into the optimization problem. Additionally, **ROI** has fewer dependencies and, last but not least, the authors are familiar with package **ROI**.

```
R> library("holiglm")
```

#### 4.1. Model fitting

Function `hglm` is the main function for fitting HGLMs within package **holiglm**.

```
hglm(formula, family = gaussian(), data, constraints = NULL,
      weights = NULL, scaler = c("auto", "center_standardization",
                                "center_minmax", "standardization", "minmax", "off"),
      scale_response = NULL, big_m = 100, solver = "auto", control = list(),
      dry_run = FALSE, object_size = c("normal", "big"))
```

The design of the function arguments resembles that of the `stats::glm()` function, with some additional arguments.

**constraints** The constraints imposed on the GLM. All constraints are of class `"hglmc"`, so the `constraints` argument expects a list of `"hglmc"` constraints.

**scaler** Especially for constraints which rely on the big- $M$  formulation, scaling is very important as it helps to make reliable choices in regard to the chosen big- $M$  value.

**scale\_response** Whether the response shall be standardized. Scaling the response is only used for family `gaussian()`, where the default is also to scale it.

**big\_m** An upper bound  $M$  for the coefficients, needed for the big- $M$  constraint.

**solver** By default, the best available optimization solver is chosen automatically. However, it is possible to force the use of a particular solver by providing the solver name. The argument is then passed along to `ROI::ROI_solve`.

**control** The control argument can be used to pass additional arguments along to `ROI::ROI_solve`.

**dry\_run** The `dry_run` argument allows to obtain the underlying optimization problem. This can be useful if the user wants to impose additional constraints, which are currently not implemented, directly to the **ROI** optimization object.

**object\_size** The `object_size` argument controls whether additional information such as the **ROI** solution as well as the `hglm_model` are stored in the fitted object.

By making use of the formula system provided by the `stats` package, we ensure that all the typical operations work as in `stats::glm`. We illustrate the usage of `hglm` without constraints for the log-binomial regression model. This particular family-link combination was chosen as a first example as it can be shown (by simulation) that using conic programming instead of the iteratively reweighted least squares (IRLS) employed by `glm` is not only more reliable but potentially faster (Schwendinger *et al.* 2021).

```
R> data("Caesarian", package = "lbreg")
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB,
+   binomial(link = "log"), solver = "ecos",
+   data = Caesarian, constraints = NULL)
R> fit
```

```
Call: hglm(formula = cbind(n1, n0) ~ RISK + NPLAN + ANTIB,
  binomial(link = "log"), data = Caesarian, constraints = NULL,
  solver = "ecos")
```

Coefficients:

| (Intercept) | RISK  | NPLAN | ANTIB  |
|-------------|-------|-------|--------|
| -1.977      | 1.295 | 0.545 | -2.066 |

Degrees of Freedom: 6 Total (i.e. Null); 3 Residual

Null Deviance: 83.49

Residual Deviance: 6.308 AIC: 31.49

Given that most of the auxiliary functions used in context of GLMs are implemented in R as S3 generics, package `holiglm` also relies on S3 generics to provide these functions. An overview on all the generics implemented in package `stats` for class `"glm"` can be obtained by:

```
R> grep("^[:alnum:]*\\.glm$", ls(getNamespace("stats")), value = TRUE)
```

```
[1] "add1.glm"      "anova.glm"      "confint.glm"
[4] "deviance.glm" "drop1.glm"      "effects.glm"
[7] "extractAIC.glm" "family.glm"     "formula.glm"
[10] "influence.glm" "logLik.glm"     "nobs.glm"
[13] "predict.glm"   "print.glm"      "residuals.glm"
[16] "rstandard.glm" "rstudent.glm"   "sigma.glm"
[19] "summary.glm"   "vcov.glm"       "weights.glm"
```

Since the `"hglm"` object inherits from class `"glm"` all the generics shown above will give similar or identical results as for a `"glm"` object. The output below showcases the similarity of the output of the `summary` function for `"hglm"` objects and `stats::glm` objects:

```
R> summary(fit)
```

Call:

```
hglm(formula = cbind(n1, n0) ~ RISK + NPLAN + ANTIB,
```

```
binomial(link = "log"), data = Caesarian, constraints = NULL,
solver = "ecos")
```

Deviance Residuals:

```
      1      2      3      4      5      6      8
1.06853 -0.34543 -2.21588  0.20040 -0.26605 -0.15022  0.05687
```

Coefficients:

```
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.9774      0.3311  -5.972 2.35e-09 ***
RISK         1.2950      0.3407   3.801 0.000144 ***
NPLAN        0.5450      0.1451   3.755 0.000174 ***
ANTIB       -2.0659      0.2814  -7.342 2.11e-13 ***
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 83.4914 on 6 degrees of freedom
Residual deviance: 6.3079 on 3 degrees of freedom
AIC: 31.489
```

Number of iterations: 26

In the above, the number of iterations is the one needed by the solver to reach convergence. Note that the `summary` function also returns the typical coefficients table which contains information on the standard errors of the coefficients, the  $z$  score and the  $p$  value corresponding to the hypothesis test  $H_0 : \beta_j = 0$  vs.  $H_1 : \beta_j \neq 0$ . The user should employ these values with care, as in the case of constrained models, the usual asymptotic theory does not necessarily hold. More information is provided in Section 4.4.

Instead of straight using function `hglm` for estimating the model, it is also possible to take a different approach, especially if the modeler wants to access the underlying optimization problem directly:

1. Generate the model with `hglm_model`.
2. Generate the underlying optimization problem with `as.OP`.
3. Fit the model with `hglm_fit`.

To generate the model, the function `hglm_model` can be used:

```
R> x <- model.matrix(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, data = Caesarian)
R> model <- hglm_model(x = x, y = with(Caesarian, cbind(n1, n0)),
+                   binomial(link = "log"))
```

Alternatively, `hglm` with the argument `dry_run` set to `TRUE` can be used to obtain the optimization model:

```
R> model <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, data = Caesarian,
+               binomial(link = "log"), constraints = NULL, dry_run = TRUE)
```

After the model is set up, the underlying optimization problem can be built by using the generic function `as.OP` exported from **ROI** and solved using function `ROI_solve` in **ROI**.

```
R> op <- as.OP(model)
R> print(op)
```

ROI Optimization Problem:

Minimize a linear objective function of length 18 with  
- 18 continuous objective variables,

subject to  
- 49 constraints of type conic.  
  |- 42 conic constraints of type 'expp'  
  |- 7 conic constraints of type 'nonneg'  
- 18 lower and 0 upper non-standard variable bounds.

```
R> ROI::ROI_solve(op)
```

Optimal solution found.  
The objective value is: 1.109144e+02

Alternatively, the model can be fitted with `hglm_fit`. Here the solver output is already converted into a form similar to the output of `glm.fit`.

```
R> fit <- hglm_fit(model)
R> fit
```

Call: NULL

```
Coefficients:
(Intercept)      RISK      NPLAN      ANTIB
      -1.977      1.295      0.545     -2.066
```

```
Degrees of Freedom: 6 Total (i.e. Null); 3 Residual
Null Deviance:      83.49
Residual Deviance: 6.308      AIC: 31.49
```

## 4.2. Constraints

The most distinctive feature of the **holiglm** package is that it allows for the usage of many predefined constraints. In Section 3 the constraints were introduced. In this section we focus on the R usage of the constraints. Table 4 gives an overview of all the available constraints in **holiglm**.

| Constraint   | Category          | Description   |
|--|-------------------|---|
| <code>k_max(k)</code> <sup>a</sup>                     | Global sparsity   | Upper bound on the number of active predictors.   |
| <code>group_sparsity(vars, k = 1L)</code> <sup>a</sup> | Group sparsity    | Upper bound on the number of active predictors within a group of predictors <code>vars</code> .   |
| <code>rho_max(rho)</code> <sup>a</sup>                 | Group sparsity    | Upper bound on the strength of the pairwise correlation; for pairs which exceed this bound only one of the two variables is allowed in the model. |
| <code>group_inout(vars)</code> <sup>a</sup>            | Group sparsity    | Forces all predictors within a group are either in (all coefficients are non-zero) or out (all coefficients are zero) of the model.               |
| <code>include(vars)</code> <sup>a</sup>                | Modeler expertise | Forces the inclusion of predictors.   |
| <code>lower(kvars)</code> <sup>b</sup>                 | Bounded domains   | Forces lower bounds on coefficients.  |
| <code>upper(kvars)</code> <sup>b</sup>                 | Bounded domains   | Forces upper bounds on coefficients.  |
| <code>linear(L, dir, rhs)</code> <sup>b</sup>          | Linear constraint | Imposes linear constraints on the coefficients or on the $z$ variables from the big- $M$ constraint.  |
| <code>group_equal(vars)</code> <sup>b</sup>            | Linear constraint | Forces all predictors in the specified group to have the same coefficient.  |
| <code>sign_coherence(vars)</code> <sup>a</sup>         | Sign coherence    | Ensures that the coefficients of the specified predictors have the same sign.   |
| <code>pairwise_sign_coherence(rho)</code> <sup>a</sup> | Sign coherence    | Ensures that coefficients of predictors that exhibit pairwise correlation more than <code>rho</code> (in absolute value) have the same sign.      |

Table 4: Overview on the pre-implemented constraints. Here only the main arguments are shown, all arguments can be found in the **holiglm** manual. The constraints marked with <sup>a</sup> are of mixed-integer type, while the ones marked with <sup>b</sup> are of linear type.

### Combining constraints

Constraints can be combined by the combine operator `c()`. In the following example, we ensure that at most 5 predictors are in the final model and that the pairwise correlation of the active variables is less than or equal to 0.8 by combining the two constraints:

```
R> c(k_max(5), rho_max(0.8))
```

List of Holistic GLM Sparsity Constraints

```
[[1]]
Holistic GLM Sparsity Constraint of Type 'k_max'
List of 1
 $ k_max: int 5

[[2]]
Holistic GLM Sparsity Constraint of Type 'rho_max'
List of 4
 $ rho_max: num 0.8
 $ exclude: chr "(Intercept)"
 $ use      : chr "everything"
 $ method  : chr "pearson"
```

### *Global sparsity*

The global sparsity constraint `k_max` can be used to enforce an upper bound on the number of non-zero predictors contained in the model. The upper bound is enforced on the number of columns in the model matrix without taking into account the intercept.

```
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+ constraints = k_max(2), data = Caesarian)
R> coef(fit)
```

| (Intercept) | RISK     | NPLAN    | ANTIB     |
|-------------|----------|----------|-----------|
| -1.818324   | 1.319768 | 0.000000 | -1.774368 |

As one can see by looking at the coefficients, in contrast to other sparsity-inducing methods, the coefficients are not just close to zero but exactly zero if not included in the model. The information about the active coefficients can be obtained by using the `active_coefficients` method or the `coef` function with `type="selected"`.

```
R> active_coefficients(fit)
```

| (Intercept) | RISK | NPLAN | ANTIB |
|-------------|------|-------|-------|
| TRUE        | TRUE | FALSE | TRUE  |

```
R> coef(fit, type="selected")
```

| (Intercept) | RISK | NPLAN | ANTIB |
|-------------|------|-------|-------|
| TRUE        | TRUE | FALSE | TRUE  |

### *Group sparsity*

The group sparsity constraint can be used to restrict the number of predictors to be included from a particular subgroup. The `vars` argument is a vector containing the names of the variables for which the constraint should be applied. Note that the names of the variables

should correspond to the column names of the model matrix. As previously mentioned, a possible use case for this type of constraint is the selection of the best out of different transformations. For illustration, we employ a Poisson regression with log link to model the number of apprentices moving to Edinburgh from different counties. Data is available in package **GLMsData** (Dunn and Smyth 2018).

```
R> data("apprentice", package = "GLMsData")
```

We specify the following model with group constraints. For the variables distance `Dist` and population `Pop` also the log transformations are considered for inclusion in the model.

```
R> fo <- Apps ~ Dist + log(Dist) + Pop + log(Pop) + Urban + Locn
R> constraints <- c(group_sparsity(c("Dist", "log(Dist)")),
+   group_sparsity(c("Pop", "log(Pop)")))
R> fit <- hglm(fo, constraints = constraints, family=poisson(),
+   data = apprentice)
R> coef(fit)
```

```
(Intercept)      Dist  log(Dist)      Pop  log(Pop)
 7.51921885  0.00000000 -2.05542948  0.00000000  0.98217367
      Urban  LocnSouth  LocnWest
-0.01705048  0.50439145 -0.10157628
```

Indeed, the logarithmically transformed variables are selected for inclusion in the model.

In-out constraints implemented in `group_inout` force the coefficients of all variables in argument `vars` to be jointly zero or different than zero. An application for this type of constraints is dummy-encoded categorical features, where either all dummy variables should be included in the model or none. The following Poisson regression model is restricted to include three predictors, whereas all dummies corresponding to the variable `Locn` (location relative to Edinburgh) should be included or excluded from the model:

```
R> fo <- Apps ~ log(Dist) + log(Pop) + Urban + Locn
R> constraints <- c(k_max(3), group_inout(c("LocnSouth", "LocnWest")))
R> fit <- hglm(fo, constraints = constraints,
+   family = poisson(), data = apprentice)
```

To restrict the pairwise collinearity, we provide the constraint `rho_max` which sets an upper bound on the pairwise collinearity. For the `mtcars` data set in R we specify the following linear model, where the empirical correlation of the variables `cyl` and `disp` is more than 0.9:

```
R> fit <- hglm(mpg ~ cyl + disp + hp + drat + wt,
+   data = mtcars, constraints = rho_max(0.9))
R> coef(fit)
```

```
(Intercept)      cyl      disp      hp      drat
34.49587578 -0.76229201  0.00000000 -0.02088543  0.81771262
      wt
-2.97331332
```



The variable `disp` is excluded from the model.

### *Modeler expertise*

The function `include` can be used to force the inclusion of a specific variable in the model. This can be useful if a modeler is interested in setting an upper bound on the number of active variables while ensuring that a certain variable stays in the model. For example, the following model on the `Caesarian` data set will include two variables and one of them will be `NPLAN`:

```
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+ constraints = c(k_max(2), include("NPLAN")), data = Caesarian)
R> coef(fit)
```

| (Intercept) | RISK      | NPLAN     | ANTIB      |
|-------------|-----------|-----------|------------|
| -1.0056143  | 0.0000000 | 0.5892485 | -1.7899819 |

### *Bounded domains*

The functions `lower` and `upper` can be used to set lower and upper bounds on the value of the coefficients. This can be used, for example, to add non-negativity constraints or any other bounds on the coefficients.

```
R> constraints <- c(lower(c(RISK = 3, ANTIB = 1e-3)), upper(c(NPLAN = -1)))
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+ constraints = constraints, data = Caesarian)
```

Warning: In `hglm_fit`: Binding linear constraints detected.  
The standard errors are corrected as described in the vignettes.

```
R> coef(fit)
```

| (Intercept) | RISK    | NPLAN    | ANTIB   |
|-------------|---------|----------|---------|
| -3.67126    | 3.00000 | -1.00000 | 0.00100 |

Note that all three constraints are binding, as the accompanying warning indicates.

### *Linear constraints*

To impose linear constraints on the coefficients, the function `linear` can be used. In the following example we will enforce that  $\beta_{\text{RISK}} \leq \beta_{\text{ANTIB}}$  in the log-binomial regression model:

```
R> risk_leq_antib <- c(RISK = 1, ANTIB = -1)
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+ constraints = linear(risk_leq_antib, "<=", 0), data = Caesarian)
```

Warning: In `hglm_fit`: Binding linear constraints detected.  
The standard errors are corrected as described in the vignettes.

```
R> coef(fit)
```

```
(Intercept)      RISK      NPLAN      ANTIB
-0.95854524 -0.30446229  0.09905898 -0.30446229
```

The next example shows how to enforce that the coefficients of RISK and NPLAN sum to one, i.e.,  $\beta_{\text{RISK}} + \beta_{\text{NPLAN}} = 1$ .

```
R> risk_nplan <- c(RISK = 1, NPLAN = 1)
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+   constraints = linear(risk_nplan, "=", 1), data = Caesarian)
```

Warning: In hglm\_fit: Binding linear constraints detected.  
The standard errors are corrected as described in the vignettes.

```
R> coef(fit)
```

```
(Intercept)      RISK      NPLAN      ANTIB
-1.3047691   0.6600016   0.3399983  -1.9232425
```

```
R> coef(fit)[["RISK"]] + coef(fit)[["NPLAN"]]
```

```
[1] 1
```

To impose both constraints at the same time, we can either combine the linear constraints or use the matrix notation as shown below.

```
R> L <- rbind(c(RISK = 1, NPLAN = 0, ANTIB = -1), c(1, 1, 0))
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+   constraints = linear(L, c("<=", "=", c(0, 1)), data = Caesarian)
```

Warning: In hglm\_fit: Binding linear constraints detected.  
The standard errors are corrected as described in the vignettes.

```
R> coef(fit)
```

```
(Intercept)      RISK      NPLAN      ANTIB
-1.8274948  -0.2560338   1.2560338  -0.2560338
```

A special case of linear constraint is `group_equal`, which ensures that the coefficients of the variables in `vars` are equal:

```
R> fit <- hglm(cbind(n1, n0) ~ RISK + NPLAN + ANTIB, binomial(link = "log"),
+   constraints = group_equal(c("RISK", "NPLAN", "ANTIB")),
+   data = Caesarian)
R> coef(fit)
```

| (Intercept) | RISK       | NPLAN      | ANTIB      |
|-------------|------------|------------|------------|
| -0.9710750  | -0.1750262 | -0.1750262 | -0.1750262 |

Finally, we showcase how linear constraints can be imposed on the  $z$  variables by setting `on_big_m = TRUE` in function `linear`. This can be useful for modeling restrictions of the type “if variable  $j$  is included variable  $k$  should also be included”. An application for this type of constraints is estimating models with polynomial features, where if a higher order of the polynomial is included in the model, it is desirable that all lower orders are also included. The following linear model example uses data `heatcap` from package `GLMsData` where the relation of heat capacity for a type of acid and temperature is investigated.

```
R> data("heatcap", package = "GLMsData")
```

We consider polynomial features up to degree five:

```
R> fo <- Cp ~ poly(Temp, 5)
```

The linear constraint below corresponds to the constraint

$$z_{\text{Temp}^5} \leq z_{\text{Temp}^4} \leq z_{\text{Temp}^3} \leq z_{\text{Temp}^2} \leq z_{\text{Temp}},$$

which ensures that a lower order polynomial feature is included if all preceding (higher order) ones are included in the model.

```
R> L <- rbind(c(-1, 1, 0, 0, 0), c(0, -1, 1, 0, 0),
+           c(0, 0, -1, 1, 0), c(0, 0, 0, -1, 1))
R> colnames(L) <- colnames(model.matrix(fo, data = heatcap))[-1]
R> lin_z_c <- linear(L, rep("<=", nrow(L)), rep(0, nrow(L)), on_big_m = TRUE)
```

When additionally setting  $k_{\max} = 3$ , we see that all polynomial features up to degree 3 will be included in the model.

```
R> constraints <- c(k_max(3), lin_z_c)
R> fit <- hglm(fo, data = heatcap, constraints = constraints)
R> coef(fit)
```

|                |                |                |                |
|----------------|----------------|----------------|----------------|
| (Intercept)    | poly(Temp, 5)1 | poly(Temp, 5)2 | poly(Temp, 5)3 |
|                | 11.2755556     | 1.8409086      | 0.3968900      |
|                |                |                | 0.1405174      |
| poly(Temp, 5)4 | poly(Temp, 5)5 |                |                |
|                | 0.0000000      | 0.0000000      |                |

### *Sign coherence constraint*

We estimate a Poisson model with identity link for the `apprentice` data where we use interactions of the location with the other predictors. In this case it can be desirable that, for each covariate, the slopes for all locations have the same sign. Note that the model includes no intercept:

```
R> fo <- Aps ~ 0 + Locn + Locn : Dist + Locn : Pop + Locn : Urban
R> constraints <- c(
+   sign_coherence(c("LocnNorth:Dist", "LocnSouth:Dist", "LocnWest:Dist")),
+   sign_coherence(c("LocnNorth:Pop", "LocnSouth:Pop", "LocnWest:Pop")),
+   sign_coherence(c("LocnNorth:Urban", "LocnSouth:Urban",
+     "LocnWest:Urban")))
R> fit <- hglm(fo, constraints = constraints, family = poisson("identity"),
+   data = apprentice)
```

Warning: In hglm\_fit: Binding linear constraints detected.  
The standard errors are corrected as described in the vignettes.

```
R> coef(fit)
```

| LocnNorth      | LocnSouth       | LocnWest        | LocnNorth:Dist |
|----------------|-----------------|-----------------|----------------|
| 4.929168e-02   | -1.325961e+01   | 1.159558e+01    | -6.226539e-03  |
| LocnSouth:Dist | LocnWest:Dist   | LocnNorth:Pop   | LocnSouth:Pop  |
| -2.172583e-01  | -9.535403e-02   | 1.458556e-01    | 1.618307e+00   |
| LocnWest:Pop   | LocnNorth:Urban | LocnSouth:Urban | LocnWest:Urban |
| 8.259307e-02   | 9.274395e-03    | 5.071724e-01    | 8.709769e-07   |

The slope parameters for distance are all negative, while the slope parameters for population and degree of urbanization are all positive.

Finally, we showcase on the `mtcars` data set how we can restrict the highly correlated predictors to have the same sign using `pairwise_sign_coherence`:

```
R> fit <- hglm(formula = mpg ~ cyl + disp + hp + drat + wt,
+   data = mtcars, constraints = pairwise_sign_coherence(0.9))
R> coef(fit)
```

| (Intercept)   | cyl           | disp          | hp            | drat         |
|---------------|---------------|---------------|---------------|--------------|
| 3.449560e+01  | -7.622500e-01 | -1.535748e-08 | -2.088604e-02 | 8.177559e-01 |
| wt            |               |               |               |              |
| -2.973326e+00 |               |               |               |              |

Unlike in the unrestricted model, variables `cyl` and `disp` both have a negative sign. Note also that, in this example, the coefficient of `disp` is close to zero, which is in line with the result obtained when using `rho_max` constraint.

### 4.3. Auxiliary functions

#### *Estimate a sequence of models*

Function `hglm_seq` can be used to estimate the sequence of models containing  $k = 1, \dots, p$  predictors. The print method for the resulting `"hglm_seq"` object includes the AIC, BIC, EBIC $_{\gamma}$  (Chen and Chen 2008), SIC (Zhu *et al.* 2020) as well as the vector of coefficients for each model. For the polynomial regression example introduced before, we can estimate the sequence of models:

```
R> fit_seq <- hglm_seq(formula = Cp ~ poly(Temp, 5), data = heatcap,
+   constraints = lin_z_c)
R> fit_seq
```

HGLM Fit Sequence:

|   | k_max          | aic            | bic            | ebic           | sic | (Intercept) | poly(Temp, 5)1 |
|---|----------------|----------------|----------------|----------------|-----|-------------|----------------|
| 1 | 5              | -64.52         | -58.29         | -57.12         | 707 | 11.27556    | 1.840907       |
| 2 | 4              | -65.60         | -60.26         | -58.27         | 698 | 11.27556    | 1.840908       |
| 3 | 3              | -64.04         | -59.59         | -57.40         | 666 | 11.27556    | 1.840909       |
| 4 | 2              | -52.30         | -48.74         | -46.74         | 543 | 11.27556    | 1.840909       |
| 5 | 1              | -24.42         | -21.75         | -20.58         | 274 | 11.27556    | 1.840909       |
|   | poly(Temp, 5)2 | poly(Temp, 5)3 | poly(Temp, 5)4 | poly(Temp, 5)5 |     |             |                |
| 1 | 0.3968896      | 0.1405173      | -0.05560878    | 0.02653112     |     |             |                |
| 2 | 0.3968899      | 0.1405173      | -0.05560884    | 0.00000000     |     |             |                |
| 3 | 0.3968900      | 0.1405174      | 0.00000000     | 0.00000000     |     |             |                |
| 4 | 0.3968900      | 0.00000000     | 0.00000000     | 0.00000000     |     |             |                |
| 5 | 0.00000000     | 0.00000000     | 0.00000000     | 0.00000000     |     |             |                |

We can observe that the model with polynomial features up to degree 4 is the one preferred by AIC, BIC and EBIC<sub>1</sub>.

#### *Group duplicates*

For binomial regression models, if all the predictors are categorical or discrete variables with few possible values, it is often the case that the model matrix contains duplicated rows. In this case, it is convenient to aggregate the original data to a data frame containing the unique values of the predictors in the rows and for each row the number of successes and failures corresponding to the response. Function `agg_binomial` can be used for this purpose. For the `Heart` data set the number of rows reduces from originally 16949 to 74:

```
R> data("Heart", package = "lbreg")
R> heart <- agg_binomial(Heart ~ ., data = Heart, as_list = FALSE)
R> c(nrow(Heart), nrow(heart))
```

```
[1] 16949    74
```

This function should always be used when possible, as it drastically reduces the size of the underlying optimization problem and therefore, reduces the solving time. After the reduction the model can be fitted as follows:

```
R> hglm(cbind(success, failure) ~ ., binomial(link = "log"),
+   data = heart, constraints = NULL)
```

#### 4.4. A note on the output of the summary function

The `summary` function in `holigm` returns information on the standard errors of the coefficients, the associated  $z$  score and the  $p$  value corresponding to the hypothesis test  $H_0 : \beta_j = 0$  vs.

$H_1 : \beta_j \neq 0$  using the normal approximation. However, in the case of constrained models, the usual asymptotic theory does not necessarily hold and the  $p$  values can be misleading in testing for significance.

We use the following strategy in computing the standard errors of the regression coefficients in **holiglm**:

- a. For the case of sparsity constraints, the standard errors in **holiglm** are computed *ex-post* using the Fisher information matrix only for the non-zero coefficients. While this practice is commonly used for model selection procedures, the reader should note that this does not take into account the uncertainty of the selection process itself. The issue of assessing significance and effect sizes from a data set after the same data set has been used to identify relevant predictors is also referred to as *selective inference* (Taylor and Tibshirani 2015). A survey of strategies which can be used to mitigate this problem is presented in Zhang, Khalili, and Asgharian (2022a). Possible approaches include adjusting the  $p$  values (see e.g., Taylor and Tibshirani 2015; Lee, Sun, Sun, and Taylor 2016), using a randomized response (Tian and Taylor 2018) or different kind of bootstrap strategies.
- b. For all other constraint types we apply a correction to the standard errors of the coefficients. More specifically, the variance-covariance matrix of the regression coefficients is computed as the corresponding portion of the covariance matrix of the parameters, Lagrangeans, and all auxiliary parameters corresponding to the optimization problem of the augmented likelihood (see e.g., Schoenberg 1997, for more computational details).

Note that the approach in b. does not distinguish between binding inequality constraints and equality constraints, however, for inference the difference is relevant. In the presence of binding inequality constraints the normal approximation may no longer valid and confidence intervals obtained by the usual  $t$  statistics can be misleading (Silvapulle and Sen 2005). For more information on constrained statistical inference we refer the readers to the monographs Robertson, Wright, and Dykstra (1988) and Silvapulle and Sen (2005), which provide a broad overview of inference under inequality constraints. In such cases, alternative strategies should be employed to derive confidence intervals for the parameters of interest (e.g., such as non-parametric or residual bootstrap). For example, Self and Liang (1987) derive large sample properties for the likelihood function in the case where the true value lies at the boundary of the parameter space, while Grömping (2010) implements methods for performing inference under linear equality and inequality constraints in normal models in the **ic.infer** R package.

In addition to performing the correction of the standard errors, the **holiglm** package returns a warning every time the employed inequality constraints are binding. This warning informs the user to proceed with caution in interpreting the output.

## 5. Comparison with existing R packages

In this section, we are showcasing the versatility of our package **holiglm** in comparison with existing R packages for fitting GLMs under linear and/or sparsity constraints.

Table 5 gives an overview of which constraints are supported by the evaluated packages **holiglm**, **abess**, **ConsReg**, **bestglm** and **restriktor**. Table 6 displays for each package which

| Package           | Constraints     |                |                 |        |                |
|-------------------|-----------------|----------------|-----------------|--------|----------------|
|                   | Global sparsity | Group sparsity | Bounded domains | Linear | Sign coherence |
| <b>holiglm</b>    | ✓               | ✓              | ✓               | ✓      | ✓              |
| <b>abess</b>      | ✓               | ✓              |                 |        |                |
| <b>bestglm</b>    | ✓               |                |                 |        |                |
| <b>ConsReg</b>    |                 |                | (✓)             | (✓)    |                |
| <b>restriktor</b> |                 |                | ✓               | ✓      |                |

Table 5: Overview of constraints supported by R packages for fitting constrained GLMs. A bracketed check mark (✓) indicates that starting values are needed for finding valid solutions to constrained problems.

| Package           | Family-link combinations |          |     |        |         |          |             |
|-------------------|--------------------------|----------|-----|--------|---------|----------|-------------|
|                   | Gaussian                 | Binomial |     |        | Poisson |          |             |
|                   | Identity                 | Logit    | Log | Probit | Log     | Identity | Square root |
| <b>holiglm</b>    | ✓                        | ✓        | ✓   | ✓      | ✓       | ✓        | ✓           |
| <b>abess</b>      | ✓                        | ✓        |     |        | ✓       |          |             |
| <b>bestglm</b>    | ✓                        | ✓        |     | ✓      | ✓       |          |             |
| <b>ConsReg</b>    | (✓)                      | (✓)      |     |        | (✓)     |          |             |
| <b>restriktor</b> | ✓                        | ✓        | (✓) | ✓      | ✓       | (✓)      | (✓)         |

Table 6: Overview of family-link combinations supported by R packages for fitting constrained GLMs. A bracketed check mark (✓) indicates that starting values are needed for finding valid solutions to constrained problems.

family-link combinations can be fitted while also making a distinction between software that is able to estimate the models directly and software that needs additional information such as, e.g., starting values. In the following, we aim to investigate by means of simulation studies how **holiglm** compares to other packages in terms of runtimes and reliability of the solution for the best subset selection problem and for non-negative GLMs. Finally, we also investigate the memory consumption of the models fitted with **holiglm** and compare it to the one of `stats::glm()` for an unconstrained problem. It should be noted that, for the various log-likelihoods, the time for solving the underlying optimization problems with **holiglm** is heavily dependent on their conic representations and the used solver. As outlined in Table 1 these representations include the second-order cone as well the primal exponential cone for linear, logistic and Poisson regression.

In setting up the exercises, we follow a simulation setting similar to the one in Hofmann, Gatu, Kontoghiorghe, Colubi, and Zeileis (2020). For generating the data of our predictor matrix  $X \in \mathbb{R}^{n \times (p+1)}$  (including an intercept term) we sample  $n$  observations from a  $p$ -dimensional multivariate normal distribution with zero mean and covariance matrix  $\Sigma := \delta_{ij}$  where  $\delta$  denotes the Kronecker delta. Given  $p$  features, we choose the first  $q = \lfloor \frac{p}{2} \rfloor$  as true features and set their respective coefficients to 1. Additionally, we set the intercept term to 1. The remaining coefficients are set to 0. Therefore, we have

$$\eta_i = 1 + x_{1i} + \dots + x_{qi}$$

together with  $g(\mu_i) = \eta_i$  where  $g$  is the link function. For the different distributions of the response, we simulated  $y_i \sim \text{Normal}(\mu_i, 1)$ ,  $y_i \sim \text{Binomial}(1, \mu_i)$  and  $y_i \sim \text{Poisson}(\mu_i)$ , respectively.

In the different exercises, we use different  $n$  and  $p$  configurations. For each configuration, we carry out 10 runs leading to 10 different data sets where we report the mean runtimes together with the standard deviation of these single runs. All computational experiments were conducted on a Dell XPS15 laptop with 32GB of RAM and an Intel Core i7-8750H CPU@2.20GHzx12 processor. As optimization solvers in **holiglm** we used **GUROBI** 9.1.2 (Gurobi Optimization LLC 2022), **ECOS** 2.0.5 (Domahidi *et al.* 2013) and **MOSEK** 10.0.34 (MOSEK ApS 2022b). For all examined examples, the open-source **ECOS** solver can be used, which is also the default solver in **holiglm**. However, regarding running times, we have also included **GUROBI** (for linear regression) and **MOSEK** (for logistic and Poisson regression) to show which runtimes are possible by using commercial solvers.

### 5.1. Fitting GLMs under global sparsity constraints

For the first study, we fit GLMs under global sparsity constraints leading to the best subset selection problem. In this setting, we compare **holiglm** with the R packages **abess** (Zhu *et al.* 2022) and **bestglm** (McLeod, Xu, and Lai 2020).

We are interested in finding models with minimal AIC as information criterion. The AIC is monotonic with respect to the log-likelihood, that is

$$\log \mathcal{L}(\beta_a; \mathbf{y}) \leq \log \mathcal{L}(\beta_b; \mathbf{y}) \Rightarrow \text{AIC}(\beta_b; \mathbf{y}) \leq \text{AIC}(\beta_a; \mathbf{y}), \quad \text{when } |\beta_a| = |\beta_b| \quad (4)$$

where  $|\beta|$  denotes the number of non-zero coefficients in  $\beta$ . In **holiglm**, we estimate the sequence of models with global sparsity constraints to find the model with minimal AIC. Out of this sequence containing the models with maximum log-likelihood for each support size, we select the model with minimal AIC. In total, this leads to solving  $p$  mixed-integer conic optimization problems.

We consider different data configurations with  $n = 1000$  observations and varying sizes of  $p = 5, 10, 15, 20, 25, 30, 35, 40$  features. For logistic and Poisson regression and linear regression with **ECOS** as solver, we only benchmark up to 20 features for the best subset selection study due to long solver running times. Moreover, since **bestglm** uses exhaustive enumeration for non-Gaussian GLMs, it is limited to data sets containing 15 features or less.

The runtimes results for the best subset selection problem in our first simulation are illustrated in Table 7 and Figures 4 to 6. We can observe that in this scenario **abess** is orders of magnitudes faster than the compared packages. In terms of scaling, Table 7 indicates that for linear and logistic regression **holiglm** scales better than **bestglm** with increasing dimensions.

We compare the different packages not only with respect to the runtime but also based on their accuracy in finding global AIC minima. To account for numerical rounding errors, we already consider the selection of model coefficients with the lowest AIC as success. Table 8 shows that **holiglm** (with **GUROBI** respectively **MOSEK** as solver) and **bestglm** are always able to find the model with the minimum AIC. As a result, they solve the best subset selection problem with regard to the information criteria.



| $p$ | $q$ | Linear                |                    |                         |                     | Logistic            |                     |                         |                    | Poisson                |                       |                         |                    |
|-----|-----|-----------------------|--------------------|-------------------------|---------------------|---------------------|---------------------|-------------------------|--------------------|------------------------|-----------------------|-------------------------|--------------------|
|     |     | hglm (ECOS)           | hglm (GUROBI)      | abess                   | bestglm             | hglm (ECOS)         | hglm (MOSEK)        | abess                   | bestglm            | hglm (ECOS)            | hglm (MOSEK)          | abess                   | bestglm            |
| 5   | 3   | 1.582<br>(0.0120)     | 0.200<br>(0.052)   | <b>0.003</b><br>(0.003) | 0.006<br>(0.001)    | 5.204<br>(0.421)    | 10.936<br>(0.859)   | <b>0.017</b><br>(0.012) | 0.233<br>(0.043)   | 4.279<br>(0.836)       | 13.516<br>(2.309)     | <b>0.002</b><br>(0.001) | 0.127<br>(0.006)   |
| 10  | 5   | 18.521<br>(1.184)     | 0.363<br>(0.038)   | <b>0.003</b><br>(0.002) | 0.007<br>(0.001)    | 24.905<br>(1.273)   | 36.740<br>(1.438)   | <b>0.031</b><br>(0.005) | 4.017<br>(0.125)   | 80.576<br>(19.975)     | 85.402<br>(11.169)    | <b>0.005</b><br>(0.001) | 3.799<br>(0.508)   |
| 15  | 8   | 171.352<br>(14.852)   | 0.665<br>(0.037)   | <b>0.004</b><br>(0.001) | 0.010<br>(0.001)    | 123.023<br>(16.354) | 114.181<br>(7.274)  | <b>0.053</b><br>(0.006) | 144.479<br>(1.694) | 712.555<br>(319.337)   | 485.869<br>(74.590)   | <b>0.008</b><br>(0.001) | 141.622<br>(4.684) |
| 20  | 10  | 1583.914<br>(290.721) | 1.100<br>(0.051)   | <b>0.007</b><br>(0.000) | 0.014<br>(0.002)    | 457.951<br>(74.369) | 309.437<br>(25.989) | <b>0.080</b><br>(0.004) | –                  | 4887.882<br>(3562.346) | 1824.087<br>(545.830) | <b>0.013</b><br>(0.001) | –                  |
| 25  | 13  | –                     | 2.113<br>(0.070)   | <b>0.012</b><br>(0.001) | 0.061<br>(0.004)    | –                   | –                   | –                       | –                  | –                      | –                     | –                       | –                  |
| 30  | 15  | –                     | 4.154<br>(0.162)   | <b>0.017</b><br>(0.001) | 0.528<br>(0.058)    | –                   | –                   | –                       | –                  | –                      | –                     | –                       | –                  |
| 35  | 18  | –                     | 31.394<br>(24.355) | <b>0.041</b><br>(0.027) | 12.723<br>(6.575)   | –                   | –                   | –                       | –                  | –                      | –                     | –                       | –                  |
| 40  | 20  | –                     | 93.016<br>(19.466) | <b>0.051</b><br>(0.023) | 110.752<br>(24.237) | –                   | –                   | –                       | –                  | –                      | –                     | –                       | –                  |

Table 7: Comparison of R packages on best subset selection with respect to runtime. The average runtimes are given in seconds together with the standard deviation (in parentheses) across 10 data sets.

| $p$ | $q$ | Linear      |               |       |         | Logistic    |              |       |         | Poisson     |              |       |         |
|-----|-----|-------------|---------------|-------|---------|-------------|--------------|-------|---------|-------------|--------------|-------|---------|
|     |     | hglm (ECOS) | hglm (GUROBI) | abess | bestglm | hglm (ECOS) | hglm (MOSEK) | abess | bestglm | hglm (ECOS) | hglm (MOSEK) | abess | bestglm |
| 5   | 3   | 10          | 10            | 9     | 10      | 10          | 10           | 10    | 10      | 10          | 10           | 10    | 10      |
| 10  | 5   | 10          | 10            | 10    | 10      | 10          | 10           | 10    | 10      | 10          | 10           | 8     | 10      |
| 15  | 8   | 10          | 10            | 10    | 10      | 10          | 10           | 10    | 10      | 8           | 10           | 8     | 10      |
| 20  | 10  | 10          | 10            | 9     | 10      | 10          | 10           | 10    | –       | 1           | 10           | 8     | –       |
| 25  | 13  | –           | 10            | 10    | 10      | –           | –            | –     | –       | –           | –            | –     | –       |
| 30  | 15  | –           | 10            | 10    | 10      | –           | –            | –     | –       | –           | –            | –     | –       |
| 35  | 18  | –           | 10            | 10    | 10      | –           | –            | –     | –       | –           | –            | –     | –       |
| 40  | 20  | –           | 10            | 10    | 10      | –           | –            | –     | –       | –           | –            | –     | –       |

Table 8: Comparison of R packages on best subset selection with respect to the number of times the method selected the correct coefficients of the model with minimal AIC for 10 data sets.

## 5.2. Fitting GLMs under non-negativity constraints

The second study deals with estimating non-negative GLMs, i.e., finding models under linear constraints, specifically under non-negativity constraints on the coefficients. We again consider different data configurations with  $n = 1000$  observations and varying sizes of  $p = 5, 10, 15, 20, 25, 30, 35, 40$  features. Here we compare **holiglm** with the packages **ConsReg** and **restriktor**.

In the case of linear constraints, we encountered the problem that for **ConsReg**, the default optimizer **solnp** provides results where coefficients can be negative contrary to the specified constraints. Package **ConsReg** relies on non-linear general-purpose optimization solvers. This has two main disadvantages:

- Starting values need to be provided, which must reside in the feasible region’s interior.
- General-purpose solvers are less reliable than conic optimization solvers in the sense that they are more likely to find local rather than global optima.

Hence, a general-purpose solver may signal success, but the returned solution is not a global optimum. Since conic solvers internally verify the solution using the Karush-Kuhn-Tucker conditions, they are unlikely to signal success without obtaining the global optima. We also tried to use a vector consisting of zeros as an initial solution (which lies on the boundary of the feasible region), but this also led to solutions with negative coefficients. To mitigate our problem, we start with finding a solution to the unconstrained optimization problem fitting

| $p$ | $q$ | Linear           |                         |                         |                   | Logistic         |                  |                  |                   | Poisson          |                  |                  |                   |
|-----|-----|------------------|-------------------------|-------------------------|-------------------|------------------|------------------|------------------|-------------------|------------------|------------------|------------------|-------------------|
|     |     | hglm (ECOS)      | hglm (GUROBI)           | ConsReg                 | restriktor        | hglm (ECOS)      | hglm (MOSEK)     | ConsReg          | restriktor        | hglm (ECOS)      | hglm (MOSEK)     | ConsReg          | restriktor        |
| 5   | 3   | 0.110<br>(0.007) | 0.069<br>(0.020)        | <b>0.032</b><br>(0.038) | 3.327<br>(0.116)  | 0.297<br>(0.013) | 0.337<br>(0.026) | 0.086<br>(0.007) | 3.426<br>(0.062)  | 0.242<br>(0.021) | 0.207<br>(0.010) | 0.074<br>(0.039) | 3.186<br>(0.092)  |
| 10  | 5   | 0.178<br>(0.008) | <b>0.055</b><br>(0.010) | 0.056<br>(0.021)        | 4.001<br>(0.058)  | 0.376<br>(0.008) | 0.419<br>(0.013) | 0.190<br>(0.161) | 4.091<br>(0.108)  | 0.346<br>(0.031) | 0.262<br>(0.023) | 0.338<br>(0.222) | 3.829<br>(0.080)  |
| 15  | 8   | 0.237<br>(0.007) | <b>0.070</b><br>(0.019) | 0.072<br>(0.033)        | 4.847<br>(0.083)  | 0.467<br>(0.042) | 0.472<br>(0.009) | 0.299<br>(0.120) | 5.009<br>(0.091)  | 0.526<br>(0.053) | 0.312<br>(0.027) | 0.744<br>(0.365) | 4.698<br>(0.086)  |
| 20  | 10  | 0.333<br>(0.031) | <b>0.076</b><br>(0.023) | 0.130<br>(0.075)        | 7.155<br>(0.378)  | 0.541<br>(0.019) | 0.584<br>(0.057) | 0.815<br>(0.776) | 6.412<br>(0.126)  | 1.320<br>(1.256) | 0.359<br>(0.009) | 1.317<br>(0.771) | 6.122<br>(0.170)  |
| 25  | 13  | 0.417<br>(0.092) | <b>0.070</b><br>(0.027) | 0.166<br>(0.084)        | 9.608<br>(2.516)  | 0.617<br>(0.015) | 0.615<br>(0.014) | 0.748<br>(0.089) | 8.790<br>(1.928)  | 1.942<br>(1.580) | 0.399<br>(0.007) | 1.990<br>(0.302) | 8.014<br>(0.191)  |
| 30  | 15  | 0.440<br>(0.031) | <b>0.068</b><br>(0.010) | 0.186<br>(0.137)        | 11.583<br>(0.478) | 0.730<br>(0.049) | 0.718<br>(0.054) | 1.147<br>(0.569) | 11.639<br>(2.020) | 4.585<br>(1.515) | 0.454<br>(0.019) | 2.343<br>(1.280) | 10.326<br>(0.328) |
| 35  | 18  | 0.471<br>(0.014) | <b>0.070</b><br>(0.006) | 0.233<br>(0.149)        | 15.384<br>(0.533) | 0.805<br>(0.047) | 0.792<br>(0.131) | 1.529<br>(0.645) | 16.185<br>(4.076) | 4.839<br>(2.170) | 0.617<br>(0.169) | 3.960<br>(1.211) | 15.564<br>(1.212) |
| 40  | 20  | 0.553<br>(0.008) | <b>0.077</b><br>(0.002) | 0.295<br>(0.179)        | 20.641<br>(0.253) | 0.881<br>(0.043) | 0.813<br>(0.036) | 2.139<br>(0.898) | 18.163<br>(0.273) | 6.847<br>(2.800) | 0.643<br>(0.054) | 2.642<br>(2.066) | 19.139<br>(1.277) |

Table 9: Comparison of R packages on fitting linear constraints with respect to runtime. The average runtimes are given in seconds together with the standard deviation (in parentheses) across 10 data sets.

a full model. We set the respective negative coefficients of this full model to zero and use it as the initial solution to `solnp`. This procedure helped increase the number of times where **ConsReg** could find a valid solution satisfying the non-negativity constraints. The time for fitting this full model is also included in the reported runtimes of package **ConsReg**.

The results presented in Table 9 and Figure 7 show that when 10 or more features are used, the **hglm** method is faster than other methods for fitting non-negative GLMs with Gaussian responses. This is achieved by using GUROBI as the optimization solver. Moreover, the timings imply that, in this case, **holiglm** scales better with higher dimensions. Table 10 displays the number of times each method was able to obtain a valid solution. It is worth noting that all methods were able to fit non-negative regression models in the case of linear regression.

For the case of logistic regression, both Table 9 and Figure 8 show that **holiglm** is faster than other packages for higher dimensions (i.e., 20 or more covariates). This is true for both conic solvers **ECOS** and **MOSEK** and we observe better scaling behavior as the dimensions increase. However, unlike linear regression, there are cases where **ConsReg** cannot find a valid solution, as indicated in Table 10.

Table 9 summarizes the runtimes for Poisson regression, and Figure 9 shows them visually. Using **ECOS** as solver the runtimes of **ConsReg** and **holiglm** are similar for lower dimensions, but **ConsReg** becomes faster when we reach 30 or more dimensions. However, Table 10 indicates that **ConsReg** cannot always find non-negative Poisson regression models. The most efficient method appears to be using **hglm** in combination with **MOSEK** as the solver. Moreover, the benchmarks indicate that this combination also has the best scaling behavior for the Poisson regression case.

In all evaluated scenarios **restriktor** is the slowest at fitting non-negative GLMs.

### 5.3. Memory usage in unconstrained logistic regression

Finally, we take a look at the used memory of models produced by **holiglm** with regards to different data sizes in an unconstrained logistic regression. Both methods for fitting **hglm** and **hglm\_fit** contain the argument `object_size`, which can take the values "normal" and "big". Notably, the "big" object is primarily intended for development purposes and includes additional components such as `hglm_model` and the **ROI** optimization problem. Figure 1 displays

| p  | q  | Linear      |               |         |           | Logistic    |              |         |           | Poisson     |              |         |           |
|----|----|-------------|---------------|---------|-----------|-------------|--------------|---------|-----------|-------------|--------------|---------|-----------|
|    |    | hglm (ECOS) | hglm (GUROBI) | ConsReg | restrktor | hglm (ECOS) | hglm (MOSEK) | ConsReg | restrktor | hglm (ECOS) | hglm (MOSEK) | ConsReg | restrktor |
| 5  | 3  | 10          | 10            | 10      | 10        | 10          | 10           | 10      | 10        | 10          | 10           | 10      | 10        |
| 10 | 5  | 10          | 10            | 10      | 10        | 10          | 10           | 9       | 10        | 10          | 10           | 9       | 10        |
| 15 | 8  | 10          | 10            | 10      | 10        | 10          | 10           | 8       | 10        | 10          | 10           | 7       | 10        |
| 20 | 10 | 10          | 10            | 10      | 10        | 10          | 10           | 5       | 10        | 10          | 10           | 6       | 10        |
| 25 | 13 | 10          | 10            | 10      | 10        | 10          | 10           | 7       | 10        | 10          | 10           | 1       | 10        |
| 30 | 15 | 10          | 10            | 10      | 10        | 10          | 10           | 8       | 10        | 10          | 10           | 2       | 10        |
| 35 | 18 | 10          | 10            | 10      | 10        | 10          | 10           | 8       | 10        | 10          | 10           | 1       | 10        |
| 40 | 20 | 10          | 10            | 10      | 10        | 10          | 10           | 7       | 10        | 10          | 10           | 3       | 10        |

Table 10: Comparison of R packages for fitting GLMs with linear constraints with respect to the number of times the method found a valid solution for 10 data sets.

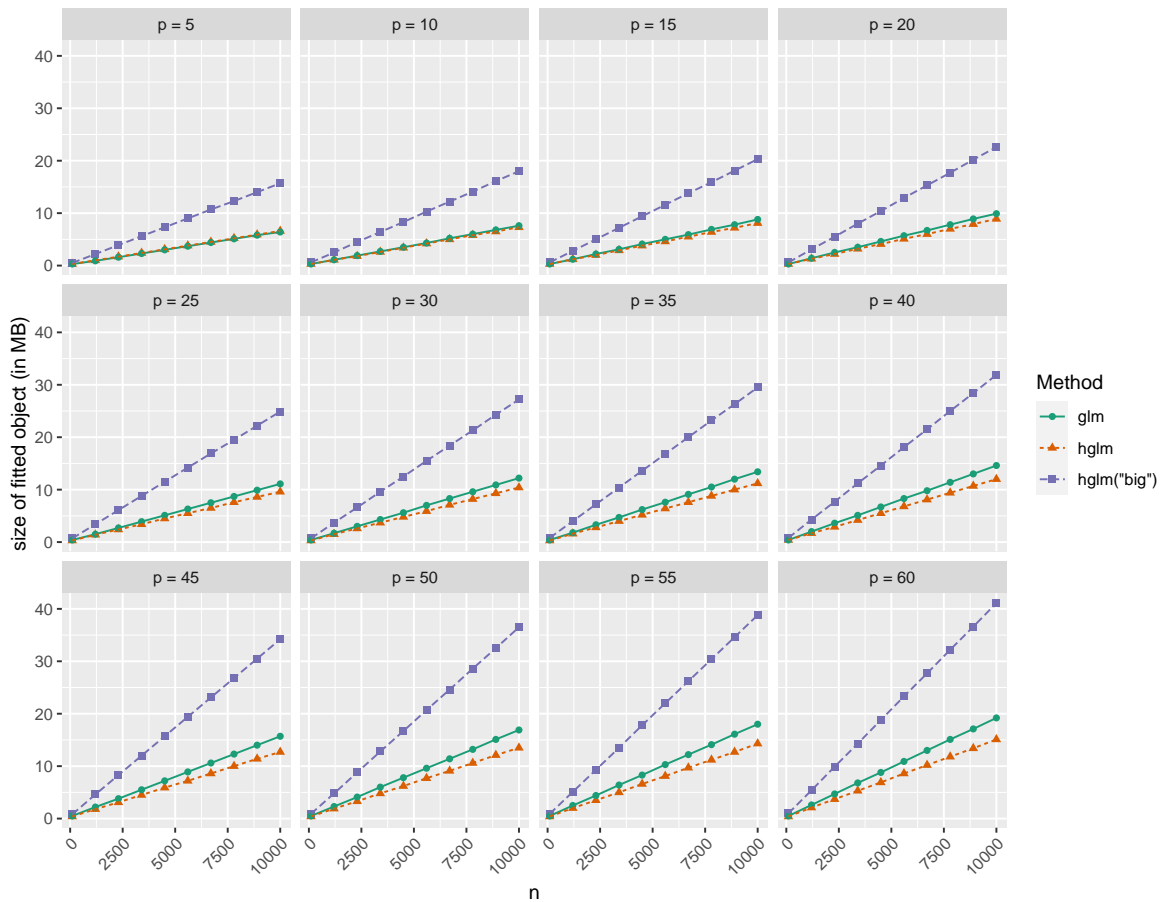


Figure 1: Comparison between `object.size` of `hglm(object_size = "normal")`, `hglm(object_size = "big")` and `glm` objects for logistic regression.

the used memory of fitted objects (both object sizes "normal" and "big") for logistic regression in comparison with plain objects of class `glm`. To provide a comprehensive evaluation, we vary the number of observations ( $n = 100, 1200, 2300, 3400, 4500, 5600, 6700, 7800, 8900, 10000$ ) and features ( $p = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$ ). Our results show that `hglm` objects with `object_size = "big"` occupy more memory than their `object_size = "normal"` counterparts while objects of `object_size = "normal"` occupy slightly less memory than comparable `glm` objects.

## 6. Use cases

This section illustrates the usage of the **holiglml** package on more realistic real-world examples.

### 6.1. Fairness in logistic regression

We illustrate how the **holiglml** package can be employed for estimating the coefficients of a logistic regression model with the fairness constraints proposed in [Zafar, Valera, Gomez-Rodriguez, and Gummadi \(2019\)](#). In the fairness literature, one is typically interested in building classification models free of the so-called *disparate impact*. A decision-making process suffers from disparate impact if it grants disproportionately large fraction of beneficial (or positive) outcomes to certain groups of a sensitive feature, such as gender or race. Assuming we have a binary response variable  $y$  and a binary sensitive feature  $w$ , a binary classifier is free of disparate impact if the probability that the classifier assigns an observation to the positive class is the same for both values of the sensitive feature  $w$ :  $\mathbb{P}(\hat{y} = 1|w = 0) = \mathbb{P}(\hat{y} = 1|w = 1)$ . If  $w$  has more than two classes, this relation should hold for all values of  $w$ . Note that even if the sensitive variable  $w$  is not included in the model, the classifier can still suffer from disparate impact (mainly due to the association between the non-sensitive variables  $\mathbf{x}$  included in the model and the sensitive variable).

However, for many classifiers these probabilities are typically non-convex functions of the parameters, which complicates estimation. To provide a more general framework, [Zafar et al. \(2019\)](#) propose a covariance measure of decision boundary unfairness. This is given by the covariance between a sensitive binary attribute  $w$  and the signed distance from the (non-sensitive) feature vectors  $\mathbf{x}_i$  to the decision boundary:

$$\text{COV}(w, d_{\beta}(\mathbf{x})) = \mathbb{E}((w - \bar{w})d_{\beta}(\mathbf{x})) - \underbrace{\mathbb{E}((w - \bar{w}))\mathbb{E}(d_{\beta}(\mathbf{x}))}_{=0} \approx \frac{1}{n} \sum_{i=1}^n (w_i - \bar{w})d_{\beta}(\mathbf{x}_i). \quad (5)$$

The framework can also accommodate for various binary sensitive variables  $w^{(k)}$ ,  $k = 1, \dots, K$ , where limits are imposed for the above covariance measure for each of the  $k$  sensitive variables. For the logistic regression problem, where the signed distance to the decision boundary is proportional to  $\mathbf{x}_i^{\top} \boldsymbol{\beta}$ , the fairness constrained model is given by:

$$\begin{aligned} & \underset{\boldsymbol{\beta}}{\text{maximize}} && \sum_{i=1}^n \log(p(y_i|\mathbf{x}_i, \boldsymbol{\beta})) \\ & \text{subject to} && \frac{1}{n} \sum_{i=1}^n (w_i^{(1)} - \bar{w}^{(1)}) \mathbf{x}_i^{\top} \boldsymbol{\beta} \leq c_1 \\ & && \frac{1}{n} \sum_{i=1}^n (w_i^{(1)} - \bar{w}^{(1)}) \mathbf{x}_i^{\top} \boldsymbol{\beta} \geq -c_1, \\ & && \dots \\ & && \frac{1}{n} \sum_{i=1}^n (w_i^{(K)} - \bar{w}^{(K)}) \mathbf{x}_i^{\top} \boldsymbol{\beta} \leq c_K \\ & && \frac{1}{n} \sum_{i=1}^n (w_i^{(K)} - \bar{w}^{(K)}) \mathbf{x}_i^{\top} \boldsymbol{\beta} \geq -c_K, \end{aligned} \quad (6)$$

where  $c_k \in \mathbb{R}^+$  for  $k = 1, \dots, K$  is a given constant which trades off accuracy and decision boundary unfairness.

As an illustrative example we employ the **AdultUCI** data set in package **arules** ([Hahsler, Buchta, Grün, and Hornik 2021](#)), which contains data extracted from the 1994 U.S. census bureau database. The response variable in this data set is the binary variable **income** which indicates whether the income of the respondents exceeds USD 50 000 per year.

```
R> data("AdultUCI", package = "arules")
```

Before estimation, we eliminate all observations with missing values:

```
R> AdultUCI <- na.omit(AdultUCI)
R> dim(AdultUCI)
```

```
[1] 30162    15
```

The data is rather unbalanced in terms of the distribution of the response:

```
R> prop.table(table(AdultUCI$income))
```

```
      small      large
0.7510775 0.2489225
```

Moreover, this data set is known to be skewed in terms of race and gender.

```
R> xtabs(~ sex + race, data = AdultUCI)
```

```
      race
sex   Amer-Indian-Eskimo Asian-Pac-Islander Black Other White
Female          107          294 1399    87 7895
Male           179          601 1418   144 18038
```

Given the sparsely populated race groups Amer-Indian-Eskimo, Asian-Pac-Islander and Other, we merge them into the Other group.

```
R> levels(AdultUCI$race)[c(1, 2)] <- "Other"
```

Clear differences are observable in the percentage of high income earners in the sex and race groups:

```
R> aggregate(income ~ sex + race, data = AdultUCI,
+   function(x) mean(as.numeric(x) - 1))
```

```
      sex race   income
1 Female Other 0.11475410
2  Male Other 0.26731602
3 Female Black 0.06075768
4  Male Black 0.19816643
5 Female White 0.12298923
6  Male White 0.32531323
```

We will therefore consider both race and gender as sensitive variables in our use case. We create one  $w^{(k)}$  binary (i.e., dummy) variable for each factor level combination.

```
R> W <- model.matrix(~ 0 + sex:race, data = AdultUCI)
```

We consider the following predictive model for the analysis (note that the sensitive variables are not included in the model formula):

```
R> form <- "income ~ age + relationship + `marital-status` + workclass +
+ `education-num` + occupation + I(`capital-gain` - `capital-loss`) +
+ `hours-per-week`"
```

The variable `native-country` is not included in the model as it was eliminated in a preliminary stepwise selection procedure.

In order to assess the trade-off between fairness and prediction accuracy, we follow [Zafar et al. \(2019\)](#) and choose a grid of values  $\alpha_k \in [0, 1]$  which are used to compute  $c_k = \alpha_k s^{(k)}$ . Here  $s^{(k)}$  is the empirical covariance (in absolute value) between the sensitive variable  $w^{(k)}$  and the linear predictor  $\mathbf{x}_i^\top \hat{\boldsymbol{\beta}}$  from the unconstrained regression;  $s^{(k)}$  serves in this case as an upper bound on the disparate impact covariance measure. A value  $\alpha_k = 0$  represents the case where the empirical covariance in Equation 5 is restricted to zero while  $\alpha_k = 1$  represents the case with no fairness constraints. We first estimate the unconstrained model using the `stats::glm` function and then calculate  $s^{(k)}$ :

```
R> m0 <- glm(as.formula(form), family = binomial(), data = AdultUCI)
R> s <- apply(W, 2, function(w) abs(cov(w, m0$linear.predictors)))
```

The left-hand side of the constraints from Equation 6 can be specified in matrix form  $L\boldsymbol{\beta}$  where  $L$  is constructed by:

```
R> xm <- model.matrix(m0)
R> L <- t(apply(W, 2, function(w) colMeans((w - mean(w)) * xm)))
```

For a grid of  $\alpha_k$  values, we estimate the fairness constrained logistic regressions. We set the `big_m` argument to 5, which means that the standardized coefficients (see `scaler = "center_standardization"`) are constrained to be at most 5 in absolute value. A value of 5 for `big_m` is small enough to ensure a fast convergence of the algorithm and, at the same time, sufficiently large to be non-restrictive for the problem at hand (in none of the estimated models this bound was reached). For each model we store the in-sample predictions, which will be used for assessing the accuracy versus fairness trade-off.

```
R> library("holiglm")
R> K <- nrow(L)
R> alpha <- seq(0, 1, by = 0.05)
R> pred_prob_race_sex <- sapply(alpha, function(ak) {
+   ck <- ak * s
+   model_constrained <- hglm(as.formula(form),
+     family = binomial(), data = AdultUCI,
+     scaler = "center_standardization", big_m = 5,
+     constraints = c(linear(L, rep(">=", K), - ck),
+       linear(L, rep("<=", K), ck)))
+   phat <- predict(model_constrained, type = "response")
+   phat
+ })
```

For each model we compute the in-sample accuracy and area under the ROC curve (using package **pROC**, Robin *et al.* 2011). Note that we consider a cut-off probability threshold of 0.5, i.e., a label of  $\hat{y} = 1$  will be assigned if the predicted probability exceeds 50%. To assess the disparate impact of the different models we use the following measure:

$$DI = \frac{P(\hat{y} = 1 | w^{(k)} = 0)}{P(\hat{y} = 1 | w^{(k)} = 1)}.$$

We only consider the dummy variable  $w^{(k)}$  which corresponds to the `White:Male` factor combination when computing  $DI$  but other approaches which take into account all  $w^{(k)}$ 's are available. The  $DI$  measure considers the `White:Male` as the privileged group in this data set and a value less than one indicates that, in the prediction, the privileged group is favored. As a rule of thumb, values greater than 0.8 can be considered acceptable.<sup>1</sup>

```
R> obs <- AdultUCI$income
R> res <- apply(pred_prob_race_sex, 2, function(phat) {
+   tab <- table(phat > 0.5, obs)
+   tab_w <- table(phat > 0.5, W[, "sexMale:raceWhite"])
+   ptab_w <- prop.table(tab_w)
+   DI <- ptab_w[2, 1]/ptab_w[2, 2]
+   acc <- sum(diag(tab))/sum(tab)
+   c(DI = DI, acc = acc, auc = pROC::roc(obs, phat, quiet = TRUE)$auc)
+ })
```

We create the scatterplots of the three measures for the grid of  $\alpha$  values:

```
R> par(mfrow = c(1, 3))
R> plot(alpha, res[1,], type = "b",
+   xlab = expression(alpha), ylab = "Disparate impact")
R> plot(alpha, res[2,], type = "b",
+   xlab = expression(alpha), ylab = "Accuracy")
R> plot(alpha, res[3,], type = "b",
+   xlab = expression(alpha), ylab = "AUROC")
```

The resulting plot can be seen in Figure 2. We observe that the model with  $\alpha = 0$  has a  $DI$  measure of around 0.6, so it still exhibits disparate impact in favor of the privileged group, even if the covariance measure is restricted to zero in the model building phase. To get a more accurate picture, one could repeat the analysis by computing the above measures on a test set. The cut-off values for the prediction can also be chosen differently. Finally, one can see that the trade-off between fairness and predictive power is not very high, as the loss in the predictive measures is not dramatic as  $\alpha_k$  decreases. Such trade-off analysis can help the modelers and decision-makers in choosing a setting which is acceptable for the use-case at hand.

---

<sup>1</sup>See also e.g., the guidelines of the US Equal Employment Opportunity Commission regarding hiring practices at <https://www.eeoc.gov/laws/guidance/questions-and-answers-clarify-and-provide-common-interpretation-uniform-guidelines>.

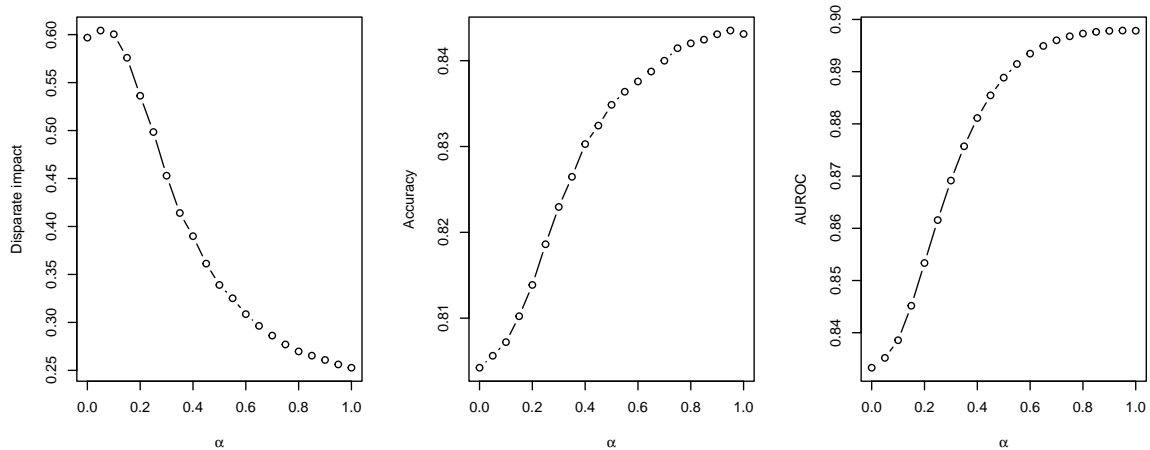


Figure 2: This figure displays the disparate impact, accuracy and area under the ROC curve for different degrees of fairness given by  $\alpha \in [0, 1]$ , where an  $\alpha$  of zero implies a fair model in terms of the specified covariance measure.

## 6.2. Model selection in log-binomial regression

In the following example we show how the **holiglm** package can be used for model selection for log-binomial regression. For this example we use the **icu** data set from the **aplore3** package (Braglia 2016). In this application, the goal is to model the status of patients at hospital discharge (i.e., alive versus dead) using a series of patient-specific predictors.

```
R> data("icu", package = "aplore3")
R> dim(icu)
```

```
[1] 200 21
```

First, we will check if the MLE exists by making use of the **detectseparation** package. For log-binomial regression the existence criterion is slightly different from the existence criterion for logistic regression, for more information we refer to Schwendinger *et al.* (2021).

```
R> library("detectseparation")
R> icu <- icu[, setdiff(colnames(icu), "id")]
R> glm(sta ~ ., family = binomial("log"), data = icu,
+     method = "detect_separation")
```

```
Data separation in log-binomial models does not necessarily result in infinite
estimates
```

```
Implementation: ROI | Solver: lpsolve
```

```
Separation: TRUE
```

```
Existence of maximum likelihood estimates
```

|             |        |              |           |           |
|-------------|--------|--------------|-----------|-----------|
| (Intercept) | age    | genderFemale | raceBlack | raceOther |
| 0           | 0      | 0            | 0         | 0         |
| serSurgical | canYes | crnYes       | infYes    | cprYes    |
| 0           | 0      | 0            | 0         | 0         |



```

      sys          hra      preYes typeEmergency      fraYes
      0            0          0          0          0
po2<= 60      ph< 7.25      pco> 45      bic< 18      cre> 2.0
      0            0          0          0          0
locStupor      locComa
      0            0
0: finite value, Inf: infinity, -Inf: -infinity

```

Package **detectseparation** signals that the data has the separation property; however, the estimates are still all expected to be finite for the log-binomial regression. Note that for logistic regression the MLE does not exist (i.e., has infinite components). To verify this, again **detectseparation** can be used.

```

R> glm(sta ~ ., family = binomial("logit"), data = icu,
+      method = "detect_separation")

```

```

Implementation: ROI | Solver: lpsolve
Separation: TRUE
Existence of maximum likelihood estimates
(Intercept)      age  genderFemale      raceBlack      raceOther
      0            0          0          -Inf          0
serSurgical      canYes      crnYes      infYes      cprYes
      0            0          0          0          0
      sys          hra      preYes typeEmergency      fraYes
      0            0          0          0          0
po2<= 60      ph< 7.25      pco> 45      bic< 18      cre> 2.0
      0            0          0          0          0
locStupor      locComa
      Inf          0
0: finite value, Inf: infinity, -Inf: -infinity

```

For the purpose of model selection, we estimate a sequence of models of different size  $k = 1, \dots, 21$  using function `hgml_seq`. In Figure 3 we present, for each model size  $k$ , the AIC and BIC of the optimized model together with the selected coefficients (which can be interpreted in terms of relative risks for the log-binomial regression).

```

R> fits <- hgml_seq(formula = sta ~ ., family = binomial("log"),
+      data = icu, solver = "ecos")

```

In terms of AIC, models of size 7–9 are preferred, while the size suggested by BIC is around 3–5. Looking at the estimated set of coefficients for each  $k$  can provide additional insights into the model behavior.

For example, the modeler can look at the stability of the magnitude of the coefficients and how they change when the number of predictors decreases. Big changes in the magnitude of the coefficients or sign changes can be an indication of multicollinearity problems within the data.

| k  | aic    | bic    | $RR_0$ | age  | genderFemale | raceBlack | raceOther | serSurgical | canYes | crnYes | infYes | epiYes | sys  | hra  | preYes | typeEmergency | fmYes | po2 ≤ 60 | ph < 7.25 | peo > 45 | bic < 18 | cre > 2.0 | locStupor | locComa |  |
|----|--------|--------|--------|------|--------------|-----------|-----------|-------------|--------|--------|--------|--------|------|------|--------|---------------|-------|----------|-----------|----------|----------|-----------|-----------|---------|--|
| 21 | 172.73 | 245.29 | 0.00   | 1.03 | 0.68         | 0.53      | 1.40      | 0.93        | 5.88   | 1.04   | 0.99   | 1.53   | 1.00 | 1.00 | 2.20   | 8.71          | 1.67  | 1.38     | 2.52      | 0.37     | 1.26     | 0.76      | 4.40      | 4.83    |  |
| 20 | 170.73 | 239.99 | 0.00   | 1.03 | 0.69         | 0.53      | 1.41      | 0.93        | 5.87   | 1.04   |        | 1.52   | 1.00 | 1.00 | 2.19   | 8.69          | 1.67  | 1.38     | 2.53      | 0.37     | 1.25     | 0.76      | 4.36      | 4.83    |  |
| 19 | 168.74 | 234.71 | 0.00   | 1.03 | 0.68         | 0.54      | 1.44      | 0.93        | 5.80   |        |        | 1.48   | 1.00 | 1.00 | 2.16   | 8.71          | 1.69  | 1.37     | 2.61      | 0.37     | 1.23     | 0.77      | 4.33      | 4.96    |  |
| 18 | 166.78 | 229.45 | 0.00   | 1.03 | 0.69         | 0.56      | 1.34      |             | 5.90   |        |        | 1.48   | 1.00 | 1.00 | 2.13   | 9.15          | 1.59  | 1.40     | 2.70      | 0.36     | 1.24     | 0.77      | 4.46      | 5.11    |  |
| 17 | 164.90 | 224.27 | 0.00   | 1.03 | 0.66         | 0.56      | 1.38      |             | 6.00   |        |        | 1.54   |      | 1.00 | 2.06   | 9.52          | 1.46  | 1.19     | 2.44      | 0.40     | 1.28     | 0.81      | 4.09      | 4.64    |  |
| 16 | 163.11 | 219.18 | 0.00   | 1.03 | 0.69         | 0.55      | 1.49      |             | 5.71   |        |        | 1.38   |      |      | 2.02   | 8.90          | 1.38  | 1.11     | 2.37      | 0.42     | 1.32     | 0.90      | 4.26      | 4.90    |  |
| 15 | 161.23 | 214.00 | 0.00   | 1.03 | 0.67         | 0.55      | 1.51      |             | 6.17   |        |        | 1.39   |      |      | 1.99   | 9.40          | 1.39  |          | 2.24      | 0.43     | 1.40     | 0.91      | 4.15      | 4.89    |  |
| 14 | 159.69 | 209.16 | 0.00   | 1.03 | 0.67         |           | 1.54      |             | 6.30   |        |        | 1.42   |      |      | 1.93   | 9.55          | 1.42  |          | 2.36      | 0.43     | 1.42     | 0.89      | 4.19      | 4.90    |  |
| 13 | 158.26 | 204.44 | 0.00   | 1.02 | 0.64         | 0.51      | 1.45      |             | 5.41   |        |        | 1.34   |      |      | 2.12   | 9.03          | 1.34  |          | 2.81      | 0.43     |          |           | 4.13      | 4.79    |  |
| 12 | 156.80 | 199.68 | 0.00   | 1.02 | 0.66         | 0.49      |           |             | 5.32   |        |        | 1.32   |      |      | 2.14   | 9.04          | 1.32  |          | 2.72      | 0.43     |          |           | 4.11      | 4.73    |  |
| 11 | 155.49 | 195.07 | 0.00   | 1.03 | 0.65         |           |           |             | 5.49   |        |        | 1.32   |      |      | 2.07   | 9.23          | 1.32  |          | 2.83      | 0.42     |          |           | 4.16      | 4.84    |  |
| 10 | 154.42 | 190.70 | 0.00   | 1.02 | 0.66         |           |           |             | 5.42   |        |        | 1.31   |      |      | 2.09   | 9.33          |       |          | 2.83      | 0.42     |          |           | 4.22      | 4.84    |  |
| 9  | 153.56 | 186.54 | 0.00   | 1.02 | 0.87         |           |           |             | 5.29   |        |        |        |      |      | 1.97   | 8.90          |       |          | 3.30      | 0.38     |          |           | 4.62      | 5.29    |  |
| 8  | 153.32 | 183.01 | 0.01   | 1.02 |              |           |           |             | 4.72   |        |        |        |      |      | 1.67   | 8.43          |       |          | 3.11      | 0.39     |          |           | 4.72      | 4.72    |  |
| 7  | 153.99 | 180.37 | 0.01   | 1.02 |              |           |           |             | 4.24   |        |        |        |      |      |        | 8.40          |       |          | 2.74      | 0.40     |          |           | 4.24      | 4.24    |  |
| 6  | 155.03 | 178.12 | 0.01   | 1.02 |              |           |           |             | 4.08   |        |        |        |      |      |        | 8.97          |       |          |           | 0.52     |          |           | 4.08      | 4.08    |  |
| 5  | 157.53 | 177.31 | 0.02   |      |              |           |           |             | 3.83   |        |        |        |      |      |        | 8.06          |       |          |           |          |          |           | 5.55      | 5.55    |  |
| 4  | 160.47 | 176.96 | 0.02   |      |              |           |           |             | 3.37   |        |        |        |      |      |        | 7.75          |       |          |           |          |          |           | 5.74      | 4.59    |  |
| 3  | 163.17 | 176.36 | 0.04   |      |              |           |           |             |        |        |        |        |      |      |        | 5.03          |       |          |           |          |          |           | 5.28      | 4.22    |  |
| 2  | 169.78 | 179.68 | 0.15   |      |              |           |           |             |        |        |        |        |      |      |        |               |       |          |           |          |          |           | 6.83      | 5.48    |  |
| 1  | 186.29 | 192.80 | 0.17   |      |              |           |           |             |        |        |        |        |      |      |        |               |       |          |           |          |          |           |           | 4.75    |  |

Figure 3: AIC, BIC and relative risks ( $\exp \beta_j$ ) for the `icu` data from the log-binomial regression with constraints on the maximum number of predictors  $k$ .  $RR_0$  corresponds to the baseline relative risk  $\exp \beta_0$ .

## 7. Conclusion

The **holigm** package can be used to fit generalized linear models which are optimal with respect to a set of specified constraints. The package provides different types of constraints which are relevant for statistical modeling purposes such as: (i) different requirements for sparsity (e.g., setting an upper limit on the number of predictors included in the model globally or from a group of pre-specified variables, forcing a group of predictors to jointly be included or excluded from the model), (ii) linear constraints, (iii) constraints on the pairwise correlation among predictors, (iv) constraints on the sign of the coefficients. The interface of the package is very similar to the one provided by `stats::glm`, making it accessible for users to specify the desired GLMs. Using the fact that the most common GLMs can be represented as conic programs, **holigm** leverages the tremendous advancements in the field of conic programming to provide efficient estimation of constrained GLMs. The package relies on **ROI** as the underlying infrastructure. Through the **ROI** plugins, problems with conic objectives and conic- and mixed-integer constraints such as the ones implemented in **holigm** can reliably be computed within R. Since conic optimization is an active field of research, this means that the runtime of estimating constrained GLMs with the **holigm** package will be reduced once more mature solvers or faster algorithms for conic optimization will become available. Future work includes extending the package with further family-link combinations, by finding appropriate conic representations (if possible) for e.g., the binomial family with complementary log-log link, the inverse Gaussian or the gamma families. Package **holigm** can be used for automatic model selection. However, we recommend using it complementary to the modeler’s expertise as an exploratory tool which provides guidance on the structure of the final model.

## Acknowledgments

Florian Schwendinger and Laura Vana acknowledge funding from the Austrian Science Fund (FWF) for the project “High-dimensional statistical learning: New methods to advance economic and sustainability policies” (ZK 35), jointly carried out by University of Klagenfurt, Vienna University of Economics and Business (WU), University of Salzburg, TU Wien, and the Austrian Institute of Economic Research (WIFO). The authors would also like to thank Camilla Damian for the helpful comments and suggestions on the first version of the manuscript, as well as the reviewers and section editor for valuable feedback on the manuscript.

## References

- Bertsimas D, King A (2015). “An Algorithmic Approach to Linear Regression.” *Operations Research*, **64**(1), 2–16. doi:10.1287/opre.2015.1436.
- Bertsimas D, Li ML (2020). “Scalable Holistic Linear Regression.” *Operations Research Letters*, **48**(3), 203–208. doi:10.1016/j.orl.2020.02.008.
- Boutaris T (2017). **colf**: *Constrained Optimization on Linear Function*. R package version 0.1.3, URL <https://CRAN.R-project.org/package=colf>.
- Boyd S, Vandenberghe L (2004). *Convex Optimization*. Cambridge University Press. ISBN 0521833787.
- Braglia L (2016). **aplore3**: *Datasets from Hosmer, Lemeshow and Sturdivant, “Applied Logistic Regression” (3rd Ed., 2013)*. R package version 0.9, URL <https://CRAN.R-project.org/package=aplore3>.
- Carrizosa E, Olivares-Nadal AV, Ramírez-Cobo P (2020). “Integer Constraints for Enhancing Interpretability in Linear Regression.” *SORT*, pp. 67–98. doi:10.2436/20.8080.02.95.
- Chaudhuri S, Handcock MS, Rendall MS (2006). **glmcc**: *An R Package for Generalized Linear Models Subject to Constraints*. URL <http://www.stat.washington.edu/handcock/combining>.
- Chen J, Chen Z (2008). “Extended Bayesian Information Criteria for Model Selection with Large Model Spaces.” *Biometrika*, **95**(3), 759–771. doi:10.1093/biomet/asn034.
- Czyzyk J, Mesnier MP, Moré JJ (1998). “The NEOS Server.” *IEEE Journal on Computational Science and Engineering*, **5**(3), 68 – 75. doi:10.1109/99.714603.
- Domahidi A, Chu E, Boyd S (2013). “ECOS: An SOCP Solver for Embedded Systems.” In *European Control Conference (ECC)*, pp. 3071–3076. doi:10.23919/ecc.2013.6669541.
- Dormann CF, Elith J, Bacher S, Buchmann C, Carl G, Carré G, Marquéz JRG, Gruber B, Lafourcade B, Leitão PJ, Münkemüller T, McClean C, Osborne PE, Reineking B, Schröder B, Skidmore AK, Zurell D, Lautenbach S (2013). “Collinearity: A Review of Methods to Deal with It and a Simulation Study Evaluating Their Performance.” *Ecography*, **36**(1), 27–46. doi:10.1111/j.1600-0587.2012.07348.x.
- Dunn PK, Smyth GK (2018). **GLMsData**: *Generalized Linear Model Data Sets*. R package version 1.0.0, URL <https://CRAN.R-project.org/package=GLMsData>.
- Fu A, Narasimhan B, Boyd S (2020). “**CVXR**: An R Package for Disciplined Convex Optimization.” *Journal of Statistical Software*, **94**(14), 1–34. doi:10.18637/jss.v094.i14.
- Grant M, Boyd S, Ye Y (2006). *Disciplined Convex Programming*, chapter 7, pp. 155–210. Springer-Verlag. ISBN 978-0-387-30528-8. doi:10.1007/0-387-30528-9\_7.
- Grömping U (2010). “Inference with Linear Equality and Inequality Constraints Using R: The Package **ic.infer**.” *Journal of Statistical Software*, **33**(10), 1–31. doi:10.18637/jss.v033.i10.

- Gurobi Optimization LLC (2022). **Gurobi Optimizer Reference Manual**. URL <https://www.gurobi.com>.
- Hahs-Vaughn DL (2016). *Applied Multivariate Statistical Concepts*. Routledge. doi:10.4324/9781315816685.
- Hahsler M, Buchta C, Grün B, Hornik K (2021). **arules: Mining Association Rules and Frequent Itemsets**. R package version 1.7-2, URL <https://CRAN.R-project.org/package=arules>.
- Hazimeh H, Mazumder R (2020). “Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms.” *Operations Research*, **68**(5), 1517–1537. doi:10.1287/opre.2019.1919.
- Helwig NE (2018). **CMLS: Constrained Multivariate Least Squares**. R package version 1.0-0, URL <https://CRAN.R-project.org/package=CMLS>.
- Hochreiter R, Schwendinger F (2020). **ROI.plugin.neos: NEOS Plug-in for the R Optimization Interface**. R package version 1.0-0, URL <https://CRAN.R-project.org/package=ROI.plugin.neos>.
- Hoerl AE, Kennard RW (1970). “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” *Technometrics*, **12**(1), 55–67. doi:10.1080/00401706.1970.10488634.
- Hofmann M, Gatu C, Kontoghiorghes EJ, Colubi A, Zeileis A (2020). “**lmSubsets**: Exact Variable-Subset Selection in Linear Regression for R.” *Journal of Statistical Software*, **93**(3), 1–21. doi:10.18637/jss.v093.i03.
- Hu Y, Li C, Meng K, Qin J, Yang X (2017). “Group Sparse Optimization via lp,q Regularization.” *Journal of Machine Learning Research*, **18**(30), 1–52.
- IBM ILOG (2022). “IBM ILOG **CPLEX** Optimization Studio.” URL <https://www.ibm.com/docs/en/icos>.
- Kosmidis I, Schumacher D, Schwendinger F (2022). **detectseparation: Detect and Check for Separation and Infinite Maximum Likelihood Estimates**. R package version 0.3, URL <https://CRAN.R-project.org/package=detectseparation>.
- Lawson CL, Hanson RJ (1995). *Solving Least Squares Problems*. SIAM.
- Lee JD, Sun DL, Sun Y, Taylor JE (2016). “Exact Post-Selection Inference, with Application to the Lasso.” *The Annals of Statistics*, **44**(3), 907–927. doi:10.1214/15-aos1371.
- McDonald JW, Diamond ID (1990). “On the Fitting of Generalized Linear Models with Non-negativity Parameter Constraints.” *Biometrics*, **46**(1), 201–206. doi:10.2307/2531643.
- McLeod AI, Xu C, Lai Y (2020). **bestglm: Best Subset GLM and Regression Utilities**. R package version 0.37.3, URL <https://CRAN.R-project.org/package=bestglm>.
- Miller A (2002). *Subset Selection in Regression*. Chapman & Hall/CRC. doi:10.1201/9781420035933.

- MOSEK ApS (2022a). *MOSEK Modeling Cookbook*. MOSEK ApS. URL <https://docs.mosek.com/MOSEKModelingCookbook-a4paper.pdf>.
- MOSEK ApS (2022b). *Rmosek: The R-to-MOSEK Optimization Interface*. R package version 10.0.34, URL <https://docs.mosek.com/latest/rmosek/>.
- Page E (1977). “Approximations to the Cumulative Normal Function and Its Inverse for Use on a Pocket Calculator.” *Journal of the Royal Statistical Society C*, **26**(1), 75–76. doi:10.2307/2346872.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. URL <https://www.R-project.org/>.
- Robertson T, Wright FT, Dykstra RL (1988). *Order Restricted Statistical Inference*, volume 229. John Wiley & Sons.
- Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M (2011). “pROC: An Open-Source Package for R and S+ to Analyze and Compare ROC Curves.” *BMC Bioinformatics*, **12**, 77. doi:10.1186/1471-2105-12-77.
- Sallés JP (2020). *ConsReg: Fits Regression & ARMA Models Subject to Constraints to the Coefficient*. R package version 0.1.0, URL <https://CRAN.R-project.org/package=ConsReg>.
- Schoenberg R (1997). “Constrained Maximum Likelihood.” *Computational Economics*, **10**(3), 251–266. doi:10.1023/a:1008669208700.
- Schwendinger B, Schwendinger F, Vana L (2024). *holiglm: Holistic Generalized Linear Models*. R package version 1.0.0, URL <https://CRAN.R-project.org/package=holiglm>.
- Schwendinger F, Grün B, Hornik K (2021). “A Comparison of Optimization Solvers for Log-Binomial Regression Including Conic Programming.” *Computational Statistics*, **36**(3), 1721–1754. doi:10.1007/s00180-021-01084-5.
- Self SG, Liang KY (1987). “Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests under Nonstandard Conditions.” *Journal of the American Statistical Association*, **82**(398), 605–610. doi:10.1080/01621459.1987.10478472.
- Silvapulle MJ, Sen PK (2005). *Constrained Statistical Inference: Inequality, Order and Shape Restrictions*. John Wiley & Sons. doi:10.1002/9781118165614.
- Slawski M, Hein M (2013). “Non-Negative Least Squares for High-Dimensional Linear Models: Consistency and Sparse Recovery without Regularization.” *Electronic Journal of Statistics*, **7**, 3004 – 3056. doi:10.1214/13-ejs868.
- Sun Q, Zhang H (2021). “Targeted Inference Involving High-Dimensional Data Using Nuisance Penalized Regression.” *Journal of the American Statistical Association*, **116**(535), 1472–1486. doi:10.1080/01621459.2020.1737079. PMID: 34538987.
- Taylor J, Tibshirani RJ (2015). “Statistical Learning and Selective Inference.” *Proceedings of the National Academy of Sciences of the United States of America*, **112**(25), 7629–7634. doi:10.1073/pnas.1507583112.

- Theußl S, Schwendinger F, Hornik K (2020). “**ROI**: An Extensible R Optimization Infrastructure.” *Journal of Statistical Software*, **94**. doi:10.18637/jss.v094.i15.
- Tian X, Taylor J (2018). “Selective Inference with a Randomized Response.” *The Annals of Statistics*, **46**(2), 679–710. doi:10.1214/17-aos1564.
- Tibshirani R (1996). “Regression Shrinkage and Selection via the LASSO.” *Journal of the Royal Statistical Society B*, **58**(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.
- Turlach BA, Weingessel A, Moler C (2019). **quadprog**: *Functions to Solve Quadratic Programming Problems*. R package version 1.5-8, URL <https://CRAN.R-project.org/package=quadprog>.
- Vanbrabant L (2022). **restriktor**: *Restricted Statistical Estimation and Inference for Linear Models*. R package version 0.3-400, URL <https://CRAN.R-project.org/package=restriktor>.
- Zafar MB, Valera I, Gomez-Rodriguez M, Gummadi KP (2019). “Fairness Constraints: A Flexible Approach for Fair Classification.” *Journal of Machine Learning Research*, **20**(75), 1–42. ISSN 1533-7928.
- Zhang D, Khalili A, Asgharian M (2022a). “Post-Model-Selection Inference in Linear Regression Models: An Integrated Review.” *Statistics Surveys*, **16**, 86–136. doi:10.1214/22-ss135.
- Zhang Y, Zhu J, Zhu J, Wang X (2022b). “A Splicing Approach to Best Subset of Groups Selection.” *INFORMS Journal on Computing*. doi:10.1287/ijoc.2022.1241.
- Zhu J, Wang X, Hu L, Huang J, Jiang K, Zhang Y, Lin S, Zhu J (2022). “**abess**: A Fast Best Subset Selection Library in Python and R.” *Journal of Machine Learning Research*, **23**(202), 1–7.
- Zhu J, Wen C, Zhu J, Zhang H, Wang X (2020). “A Polynomial Algorithm for Best-Subset Selection Problem.” *Proceedings of the National Academy of Sciences of the United States of America*, **117**(52), 33117–33123. doi:10.1073/pnas.2014241117.

## A. Conic formulations

In this section we reformulate the log-likelihood of the different family-link combinations in terms of Equation 1. Let  $\mathbf{x}_i^\top \boldsymbol{\beta} = \eta_i$ .

The family-link combinations implemented in **holigm** can be expressed by the zero and free cones:

$$\mathcal{K}_{\text{zero}} = \{0\}, \mathcal{K}_{\text{free}} = \mathbb{R} = \mathcal{K}_{\text{zero}}^*,$$

where  $\mathcal{K}^* = \{y \in \mathbb{R}^n \mid y^\top x \geq 0 \forall x \in \mathcal{K}\}$  denotes the dual cone for a closed convex cone  $\mathcal{K}$ , the linear cone:

$$\mathcal{K}_{\text{lin}} = \{x \in \mathbb{R} \mid x \geq 0\},$$

the second-order cone:

$$\mathcal{K}_{\text{soc}}^n = \{(t, \mathbf{x}) \in \mathbb{R}^n \mid \mathbf{x} \in \mathbb{R}^{n-1}, t \in \mathbb{R}, \|\mathbf{x}\|_2 \leq t\}$$

and the primal exponential cone:

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

### A.1. Gaussian

For the Gaussian family, the log-likelihood is proportional to:

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\beta}; y_1, \dots, y_n) &\propto \sum_{i=1}^n -\frac{(y_i - g^{-1}(\eta_i))^2}{2} \\ &\propto \sum_{i=1}^n -\frac{y_i^2 - 2y_i g^{-1}(\eta_i) + g^{-1}(\eta_i)^2}{2} \propto \sum_{i=1}^n y_i \underbrace{g^{-1}(\eta_i)}_{\lambda_i(\boldsymbol{\beta})} - \underbrace{\frac{g^{-1}(\eta_i)^2}{2}}_{b(\lambda_i(\boldsymbol{\beta}))} \end{aligned}$$

#### *Identity link*

For the identity link we have  $g^{-1}(\eta_i) = \eta_i$  so

$$\lambda_i(\boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta}, \quad b(\lambda_i(\boldsymbol{\beta})) = (\mathbf{x}_i^\top \boldsymbol{\beta})^2 / 2.$$

Using an auxiliary variable  $\zeta$ , an inequality of the form  $\zeta \geq w^2$  can be expressed by a second-order cone:

$$\zeta \geq z^2 \iff (\zeta + 1, \zeta - 1, 2w) \in \mathcal{K}_{\text{soc}}^3,$$

since

$$(\zeta + 1, \zeta - 1, 2w) \in \mathcal{K}_{\text{soc}}^3 \iff \|(\zeta - 1, 2w)\|_2 \leq \zeta + 1$$

and by using the definition of the norm and squaring both sides we get

$$(\zeta - 1)^2 + (2w)^2 \leq (\zeta + 1)^2 \iff 4z^2 \leq 4\zeta.$$

Using the same technique on  $\mathbf{w} \in \mathbb{R}^n$  leads to

$$\zeta \geq \sum_{i=1}^n z_i^2 \iff (\zeta + 1, \zeta - 1, 2z_1, \dots, 2z_n) \in \mathcal{K}_{\text{soc}}^{n+2}. \quad (7)$$

Since  $\max \sum_{i=1}^n y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \frac{(\mathbf{x}_i^\top \boldsymbol{\beta})^2}{2}$  can be rewritten into  $\max - \sum_{i=1}^n \frac{1}{2} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 - \frac{y_i^2}{2}$ , and by dropping the constant term  $\sum_{i=1}^n \frac{y_i^2}{2}$  we obtain the following reformulation:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \zeta}{\text{maximize}} && -\zeta \\ & \text{subject to} && (\zeta + 1, \zeta - 1, 2(y_1 - \mathbf{x}_1^\top \boldsymbol{\beta}), \dots, 2(y_n - \mathbf{x}_n^\top \boldsymbol{\beta})) \in \mathcal{K}_{\text{soc}}^{n+2} \\ & && \boldsymbol{\beta} \in \mathbb{R}^p, \zeta \in \mathbb{R}. \end{aligned}$$

## A.2. Binomial

For the binomial family, the log-likelihood is proportional to:

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\beta}; y_1, \dots, y_n) &\propto \sum_{i=1}^n y_i \log(g^{-1}(\eta_i)) + (1 - y_i) \log(1 - g^{-1}(\eta_i)) \\ &\propto \sum_{i=1}^n y_i \underbrace{(\log(g^{-1}(\eta_i)) - \log(1 - g^{-1}(\eta_i)))}_{\lambda_i(\boldsymbol{\beta})} - \underbrace{(-\log(1 - g^{-1}(\eta_i)))}_{b(\lambda_i(\boldsymbol{\beta}))} \end{aligned}$$

### Logit link

For the logit link we have:

$$\begin{aligned} \lambda_i(\boldsymbol{\beta}) &= \log(g^{-1}(\eta_i)) - \log(1 - g^{-1}(\eta_i)) \\ &= \log\left(\frac{\exp(\eta_i)}{1 + \exp(\eta_i)}\right) - \log\left(1 - \frac{\exp(\eta_i)}{1 + \exp(\eta_i)}\right) = \eta_i \\ b(\lambda_i(\boldsymbol{\beta})) &= -\log(1 - g^{-1}(\eta_i)) \\ &= -\log\left(1 - \frac{\exp(\eta_i)}{1 + \exp(\eta_i)}\right) = \log(1 + \exp(\eta_i)). \end{aligned}$$

Employing an auxiliary variable  $\delta$ , the inequality  $\log(1 + \exp(x))$  can be represented using the exponential cone. For introducing the representation, note that the exponential function can be modeled by an exponential cone:

$$\delta \geq \exp(x) \iff (x, 1, \delta) \in \mathcal{K}_{\text{exp}}. \quad (8)$$

Now, the inequality

$$\delta \leq \log(1 + \exp(x)) \iff \exp(\delta) \leq 1 + \exp(x) \quad (9)$$



can be reformulated as:

$$\begin{aligned} 1 + \gamma &\geq \exp(\delta) \iff (\delta, 1, 1 + \gamma) \in \mathcal{K}_{\text{exp}} \\ \gamma &\geq \exp(x) \iff (x, 1, \gamma) \in \mathcal{K}_{\text{exp}} \end{aligned}$$

The objective is therefore reformulated as:

$$\begin{aligned} &\underset{\beta, \delta, \gamma}{\text{maximize}} \sum_{i=1}^n y_i \mathbf{x}_i^\top \beta - \delta_i \\ &\text{subject to } (\delta_i, 1, 1 + \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ &\quad (\mathbf{x}_i^\top \beta, 1, \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ &\quad \beta \in \mathbb{R}^p, \delta \in \mathbb{R}^n, \gamma \in \mathbb{R}^n. \end{aligned}$$

### Log link

For the log link we have:

$$\begin{aligned} \lambda_i(\beta) &= \log(g^{-1}(\eta_i)) - \log(1 - g^{-1}(\eta_i)) \\ &= \log(\exp(\eta_i)) - \log(1 - \exp(\eta_i)) = \eta_i - \log(1 - \exp(\eta_i)) \\ b(\lambda_i(\beta)) &= -\log(1 - g^{-1}(\eta_i)) = -\log(1 - \exp(\eta_i)) \end{aligned}$$

Similar as before, the inequality  $\delta \leq \log(1 - \exp(x))$  can be represented using the exponential cone. Now

$$\delta \leq \log(1 - \exp(x)) \iff \exp(\delta) \leq 1 - \exp(x)$$

can be expressed as (using Equation 8):

$$\begin{aligned} 1 - \gamma &\geq \exp(\delta) \iff (\delta, 1, 1 - \gamma) \in \mathcal{K}_{\text{exp}} \\ \gamma &\geq \exp(x) \iff (x, 1, \gamma) \in \mathcal{K}_{\text{exp}}. \end{aligned}$$

The objective is therefore reformulated as:

$$\begin{aligned} &\underset{\beta, \delta, \gamma}{\text{maximize}} \sum_{i=1}^n y_i \mathbf{x}_i^\top \beta + (1 - y_i) \delta_i \\ &\text{subject to } \mathbf{x}_i^\top \beta \leq 0 \text{ for all } i \in \{1, \dots, n\} \\ &\quad (\delta_i, 1, 1 - \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ &\quad (\mathbf{x}_i^\top \beta, 1, \gamma_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ &\quad \beta \in \mathbb{R}^p, \delta \in \mathbb{R}^n, \gamma \in \mathbb{R}^n. \end{aligned}$$

### A.3. Poisson

For the Poisson family, the log-likelihood is proportional to:

$$\log \mathcal{L}(\beta; y_1, \dots, y_n) \propto \sum_{i=1}^n y_i \underbrace{\log(g^{-1}(\eta_i))}_{\lambda_i(\beta)} - \underbrace{g^{-1}(\eta_i)}_{b(\lambda_i(\beta))}$$

*Log link*

For the log link we have:

$$\lambda_i(\boldsymbol{\beta}) = \eta_i, \quad b(\lambda_i(\boldsymbol{\beta})) = \exp(\eta_i).$$

Using Equation 8, the objective can be reformulated as:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \boldsymbol{\delta}}{\text{maximize}} \sum_{i=1}^n y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \delta_i \\ & \text{subject to } (\mathbf{x}_i^\top \boldsymbol{\beta}, 1, \delta_i) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ & \quad \boldsymbol{\beta} \in \mathbb{R}^p, \boldsymbol{\delta} \in \mathbb{R}^n. \end{aligned}$$

*Identity link*

For the identity link we have:

$$\lambda_i(\boldsymbol{\beta}) = \log(\eta_i), \quad b(\lambda_i(\boldsymbol{\beta})) = \eta_i.$$

Imposing  $\eta_i \geq 0$  for all  $i = 1, \dots, n$  and using Equation 8, the objective can be reformulated as:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \boldsymbol{\delta}}{\text{maximize}} \sum_{i=1}^n y_i \delta_i - \mathbf{x}_i^\top \boldsymbol{\beta} \\ & \text{subject to } (\delta_i, 1, \mathbf{x}_i^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ & \quad \mathbf{x}_i^\top \boldsymbol{\beta} \geq 0 \text{ for all } i \in \{1, \dots, n\} \\ & \quad \boldsymbol{\beta} \in \mathbb{R}^p, \boldsymbol{\delta} \in \mathbb{R}^n. \end{aligned}$$

*Square root link*

For the square root link we have:

$$\lambda_i(\boldsymbol{\beta}) = 2 \log(\eta_i), \quad b(\lambda_i(\boldsymbol{\beta})) = \eta_i^2.$$

Using Equations 7 and 8, the objective can be reformulated as:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \boldsymbol{\delta}, \zeta}{\text{maximize}} \sum_{i=1}^n 2y_i \delta_i - \zeta \\ & \text{subject to } (\delta_i, 1, \mathbf{x}_i^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{exp}} \text{ for all } i \in \{1, \dots, n\} \\ & \quad (\zeta + 1, \zeta - 1, 2\mathbf{x}_1^\top \boldsymbol{\beta}, \dots, 2\mathbf{x}_n^\top \boldsymbol{\beta}) \in \mathcal{K}_{\text{soc}}^{n+2} \\ & \quad \boldsymbol{\beta} \in \mathbb{R}^p, \boldsymbol{\delta} \in \mathbb{R}^n, \zeta \in \mathbb{R}. \end{aligned}$$

## B. Runtime plots

This appendix contains the beeswarm plots for the runtime benchmarks in Section 5. The data described by these plots along with scripts to generate and plot the results can be found in the replication material of this paper.

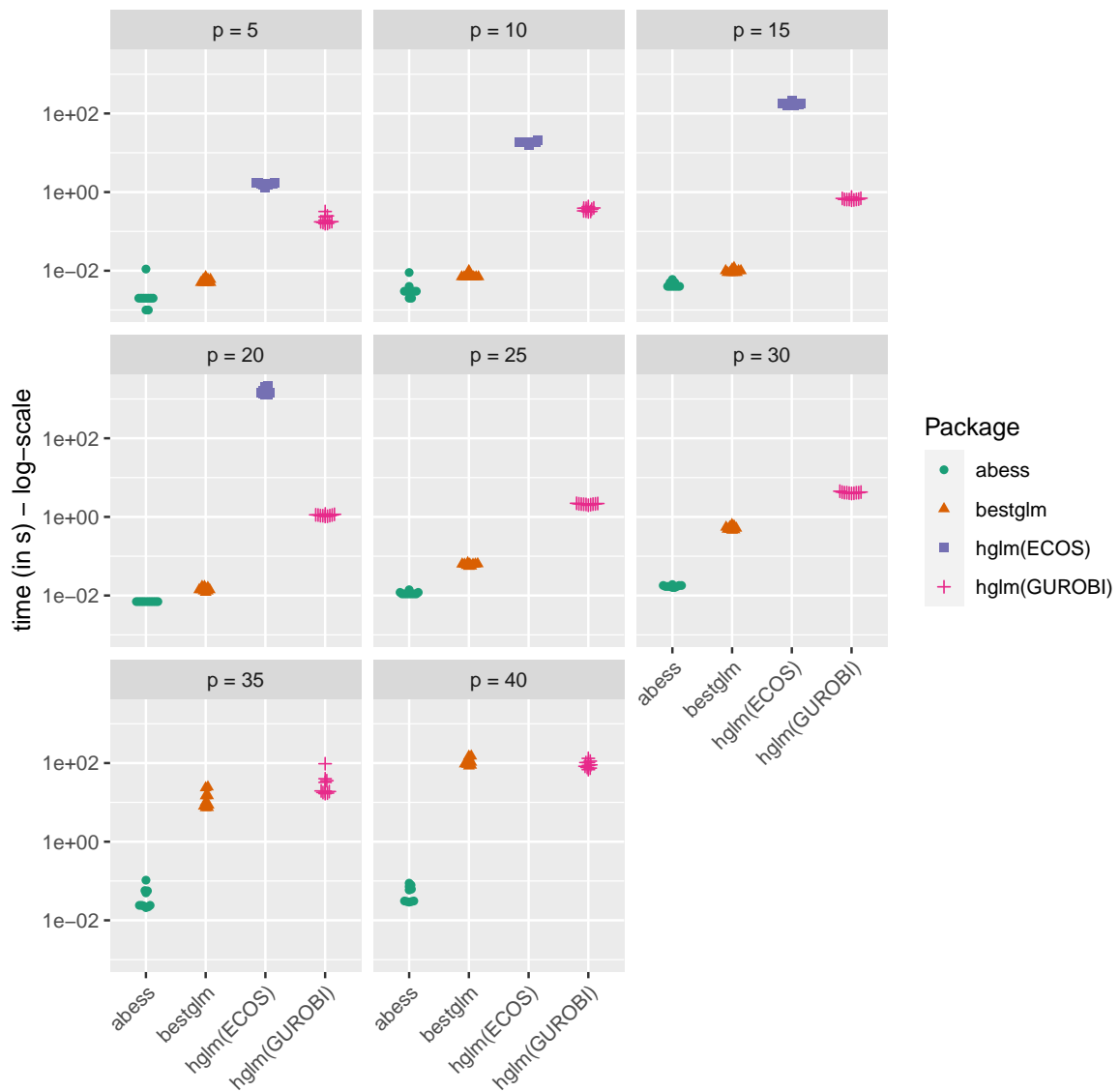


Figure 4: Runtimes for best subset selection for linear regression across 10 data sets.

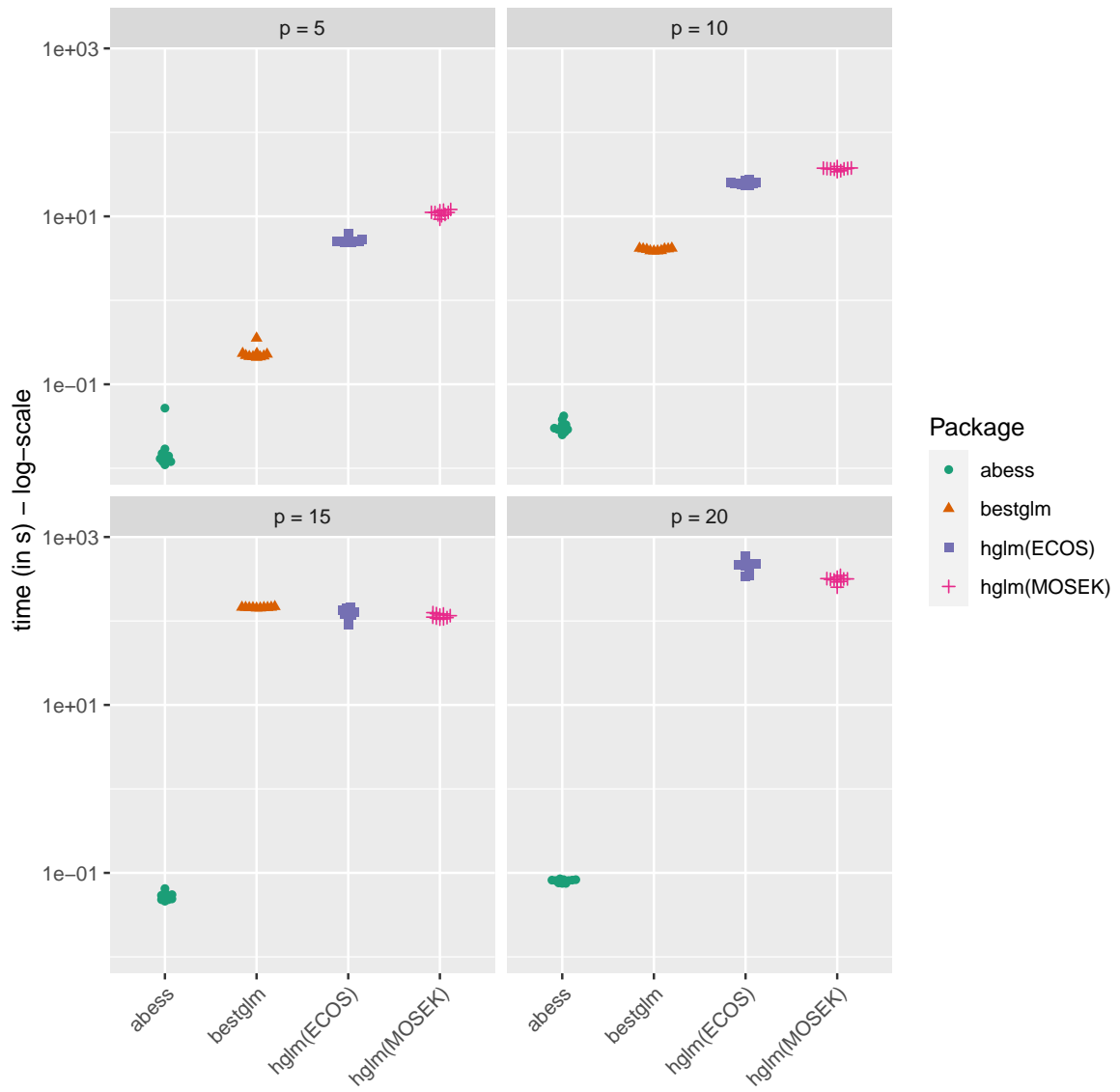


Figure 5: Runtimes for best subset selection for logistic regression across 10 data sets.

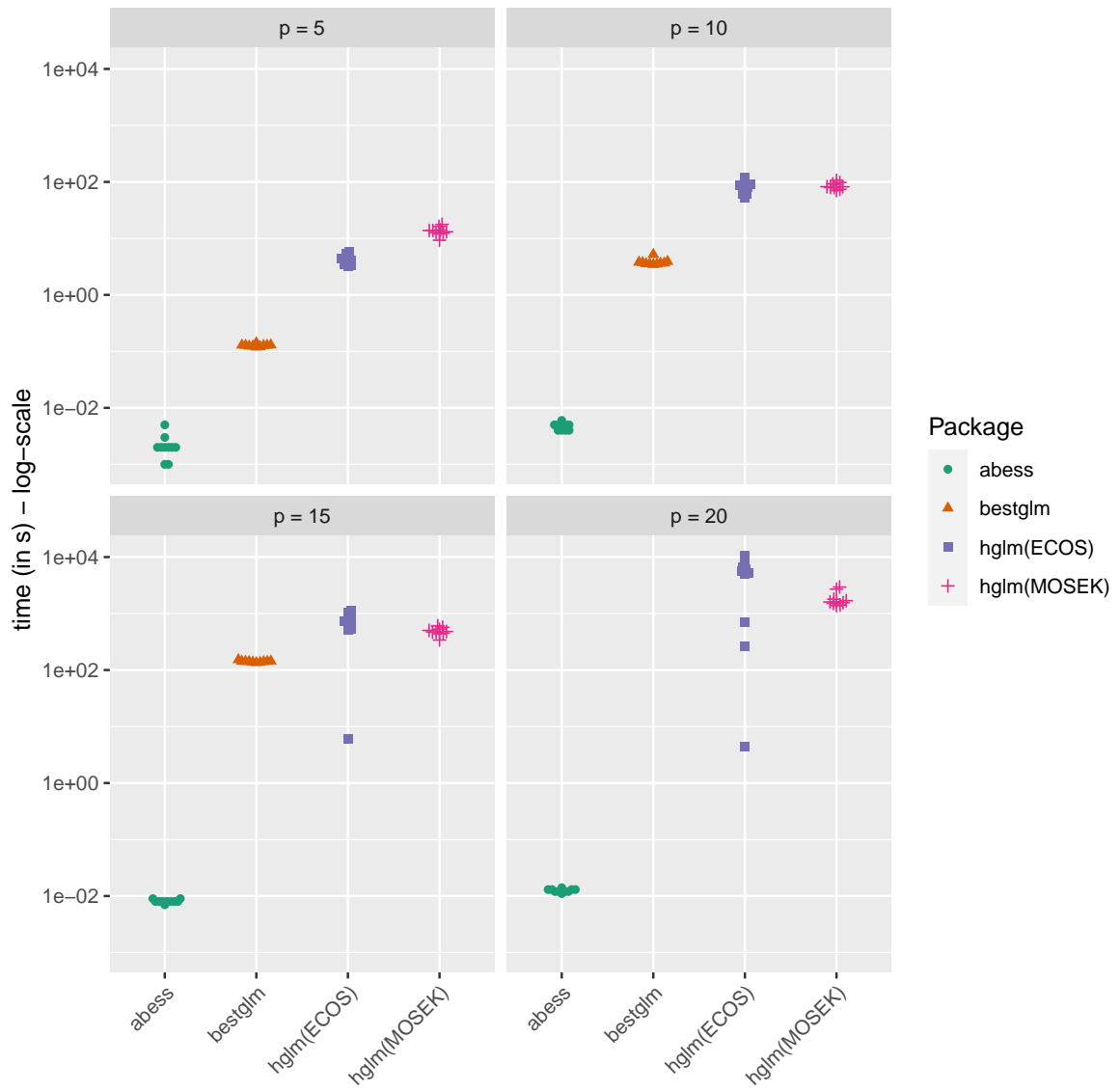


Figure 6: Runtimes for best subset selection for Poisson regression across 10 data sets.

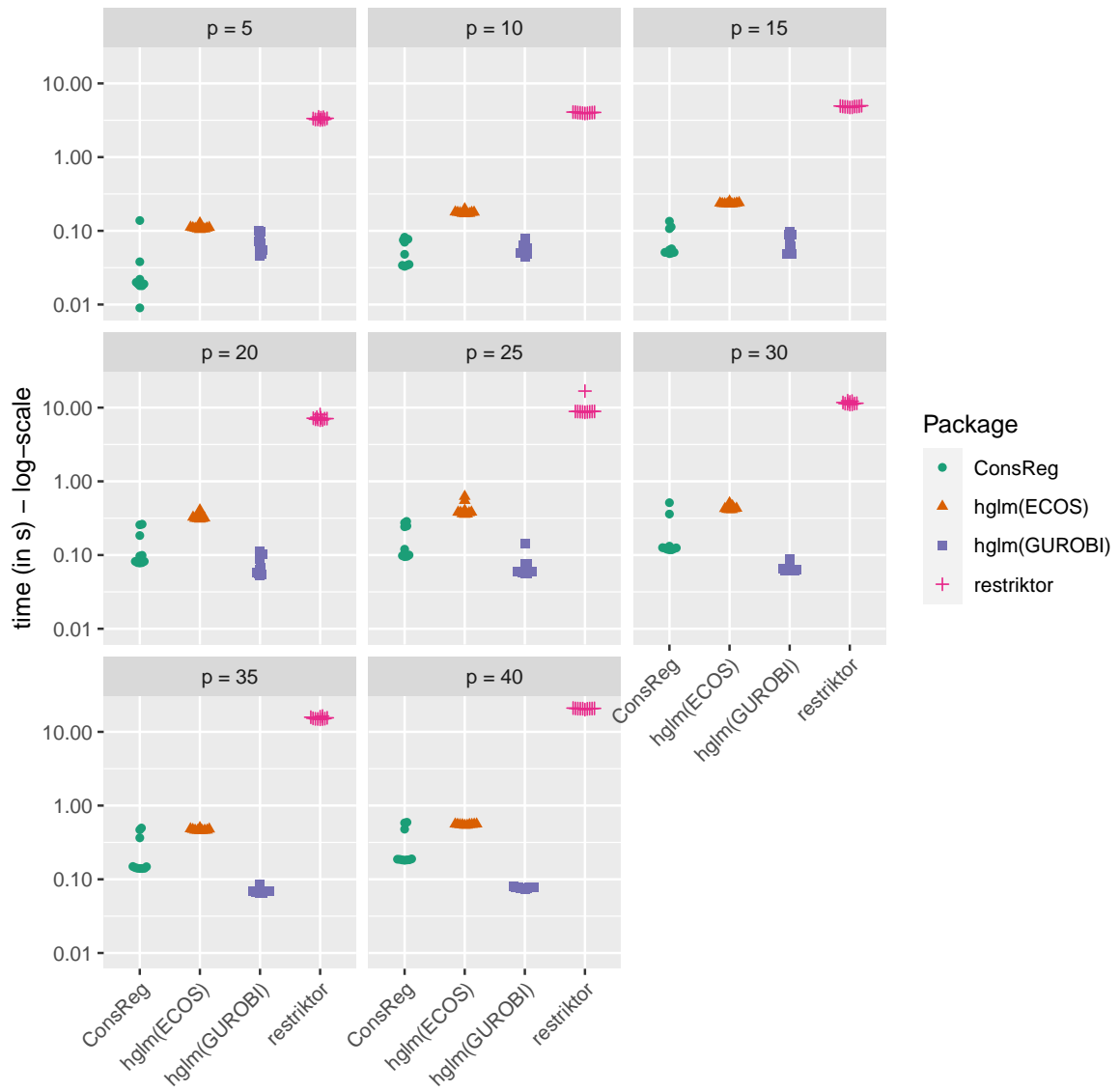


Figure 7: Runtimes for non-negative linear regression across 10 data sets.

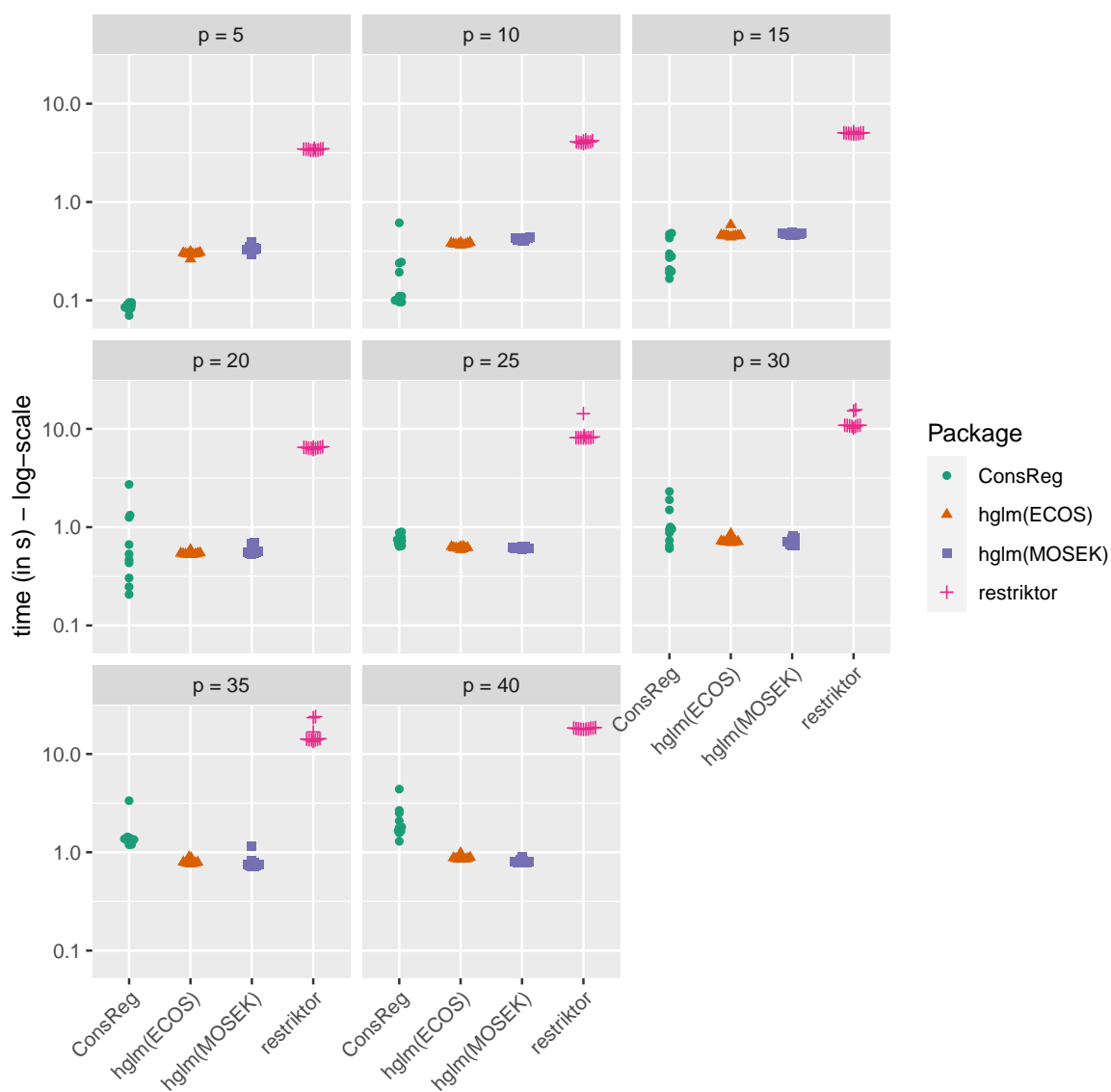


Figure 8: Runtimes for non-negative logistic regression across 10 data sets.

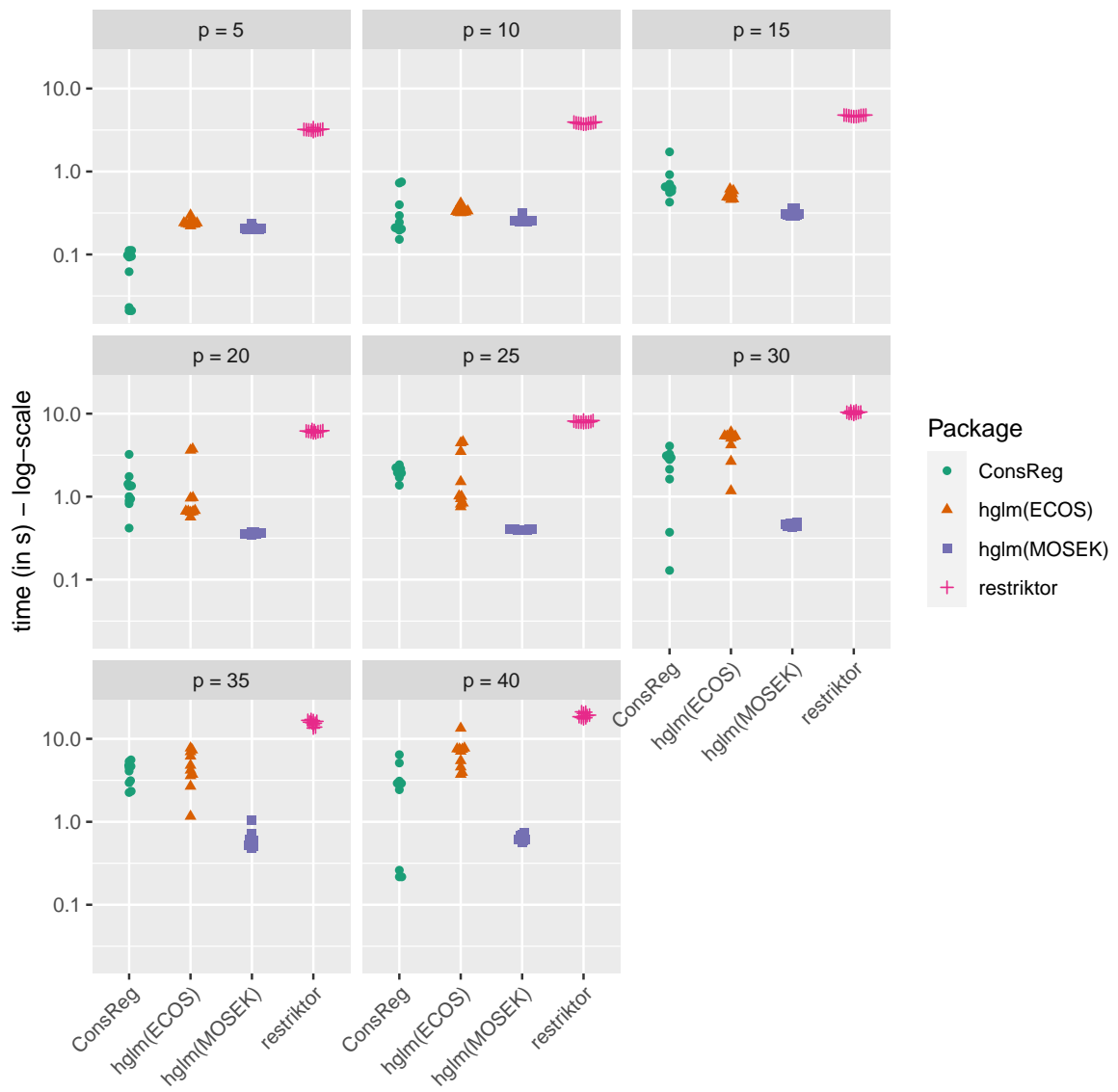


Figure 9: Runtimes for non-negative Poisson regression across 10 data sets.



**Affiliation:**

Benjamin Schwendinger  
Institute of Computer Technology  
Technische Universität Wien  
Gußhausstraße 27–29, 1040 Vienna, Austria  
E-mail: [benjaminschwe@gmail.com](mailto:benjaminschwe@gmail.com)

Florian Schwendinger  
Department of Statistics  
University of Klagenfurt  
Universitätsstraße 65–67, 9020 Klagenfurt, Austria  
E-mail: [FlorianSchwendinger@gmx.at](mailto:FlorianSchwendinger@gmx.at)

Laura Vana  
CSTAT – Computational Statistics  
Institute of Statistics and Mathematical Methods in Economics  
Technische Universität Wien  
Wiedner Hauptstraße 7, 1040 Vienna, Austria  
E-mail: [laura.vana.guer@tuwien.ac.at](mailto:laura.vana.guer@tuwien.ac.at)