LORENZ REGRESSION: AN IMPLEMENTATION OF THE LORENZ AND PENALIZED LORENZ REGRESSIONS IN R

Alexandre Jacquemain, Cédric Heuchenne

LIDAM Discussion Paper ISBA
2023 / 27

UCLouvain

LIDAM
ISBA | Louvain Institute of Data Analysis and Modeling in economics and statistics

ISBA

Voie du Roman Pays 20 - L1.04.01

B-1348 Louvain-la-Neuve

Email : lidam-library@uclouvain.be

https://uclouvain.be/en/research-institutes/lidam/isba/publication.html

LorenzRegression: an implementation of the Lorenz and penalized Lorenz regressions in `R`

Alexandre Jacquemain
UCLouvain
Cédric Heuchenne
University of Liège

Abstract

Lorenz regressions are statistical tools for measuring the extent of inequality in a response variable that is attributable to a set of covariates. These regression techniques estimate the explained Gini coefficient, a measure of inequality in the conditional expectation of the response given the covariates, assuming a single-index model. In this paper, we describe the **LorenzRegression** package, which implements the non-penalized and penalized Lorenz regressions. The non-penalized procedure is implemented via a genetic algorithm, making use of the **GA** package. For the penalized case, the user can choose between using a lasso or SCAD penalty. In the former, the estimation procedure is performed with the FABS algorithm, while the latter is implemented with the SCAD-FABS algorithm. Computationally intensive parts of the code are written in `C++`. A mathematical description of the procedure is provided. The main function is carefully described and its use is illustrated on income data.

1 Introduction

Lorenz regressions (Heuchenne and Jacquemain, 2022) are used in economics to measure the extent of inequality in a response variable Y that can be attributed to a set of covariates X. Formally, it provides inference for the explained Gini coefficient, a measure of the inequality in the conditional expectation of Y given X. A direct application arises when the covariates are circumstances beyond the individual's control. In that case, the explained Gini coefficient connects with the literature on inequality of opportunity (Jacquemain *et al.*, 2022b).

The proposed method possesses several advantages. It assumes a single-index model which, due to the presence of a nonparametric link function, is more flexible than the parametric models often used in inequality measurement. Furthemore, an estimator for the explained Gini coefficient can be obtained without having to estimate the link function. The penalized Lorenz regression (Jacquemain *et al.*, 2022a) automatically selects the relevant covariates and

avoids overestimation of the explained Gini coefficient. Both the non-penalized and penalized methods come with a proper inference procedure.

The **LorenzRegression** package is the first and only implementation so far of the Lorenz regression methodology in `R`, (R Core Team, 2021). More generally, it can be used to measure inequality, through the Gini coefficient and concentration index, and to graphically represent it, via the Lorenz and concentration curves. We refer the reader to Section 2.1 for a definition of these objects. All computations may include sample weights. Inequality measurement does not benefit from a widespread implementation in `R`, even though a few packages are publicly available from the Comprehensive `R` Archive Network (CRAN). The **laeken** package (Alfons and Templ, 2012) estimates indicators of social exclusion from complex surveys and can be used to compute Gini coefficients. The **ineq** package (Zeileis, 2014) estimates several inequality measures, including the Gini coefficient and the Lorenz curve. The **wINEQ** package (Machowska *et al.*, 2022) estimates inequality measures for weighted data, including the Gini coefficient. While we are not aware of any package available on CRAN, an open source implementation of the concentration index is proposed in the **rineq** package (Devleesschauwer *et al.*, 2017). It also comes with a parametric decomposition method, following the procedure proposed by Speybroeck *et al.* (2010).

Estimation in the Lorenz regression is based on maximizing a discrete objective function. Due to lack of differentiability, the numerical solution is obtained through a genetic algorithm, implemented via the `R` package **GA** (Scrucca, 2013). In the penalized Lorenz regression, the issue is circumvented by replacing the discrete objective function with a smooth approximation. The numerical solution is obtained via the FABS algorithm (Shi *et al.*, 2018) in the case of a lasso penalty and via the SCAD-FABS algorithm (Jacquemain *et al.*, 2022b) if the SCAD penalty is chosen instead. In the **LorenzRegression** package, the main functions are implemented in `C++` and integrated in `R`, via the **Rcpp** (Eddelbuettel and Balamuta, 2018) and **RcppArmadillo** (Eddelbuettel and Sanderson, 2014) packages. To further reduce computation time, parallel computing is implemented with the **foreach** package (Microsoft and Weston, 2022). Finally, all plots are produced with **ggplot2** (Wickham, 2016).

The paper is organized as follows. In Section 2, we introduce the model settings, the explained Gini coefficient and the Lorenz regression procedure. Section 3 focuses on the penalized case. Section 4 presents the main function of the package and the associated S3 methods. Section 5 illustrates the use of the package on a real-data example. Finally, Section 6 concludes.

2 The Lorenz regression

2.1 Model settings and the explained Gini coefficient

We consider an economic outcome $Y \in \mathbb{R}$ with $0 < \mathsf{E}[Y] < \infty$, where $\mathsf{E}[\cdot]$ is the expected value, and a vector of p covariates $X = (X_1, \ldots, X_p)^\intercal \in \mathbb{R}^p$. We start

by introducing measures of inequality. The Lorenz curve of Y evaluated at p is defined as

$$\mathrm{LC}_Y(p) = \frac{\mathsf{E}[Y \mathbb{1}\{F_Y(Y) \leq p\}]}{\mathsf{E}[Y]},$$

where $F_Y(\cdot)$ is the cumulative distribution function (CDF) of Y and $\mathbb{1}(\cdot)$ is the indicator function. It measures the share of total outcome held by the proportion p of the individuals with the smallest values of Y. In case of perfect equality, the Lorenz curve coincides with the 45° line. The Gini coefficient is twice the area between the 45° line and the Lorenz curve. It is also equivalent to the following definition

$$\mathrm{Gi}_Y = \frac{2\mathsf{COV}[Y, F_Y(Y)]}{\mathsf{E}[Y]},$$

where $\mathsf{COV}[\cdot, \cdot]$ is the covariance operator. When the ordering is produced by another variable, say X_k, the inequality measure thus obtained is called the concentration index, defined as

$$\mathrm{CI}_{Y,X_k} = \frac{2\mathsf{COV}[Y, F_{X_k}(X_k)]}{\mathsf{E}[Y]}.$$

Similarly, the Lorenz curve becomes the concentration curve.

In the context of the Lorenz regression, we assume the single-index model

$$\mathsf{E}[Y|X = x] = H(x^\intercal \theta_0), \tag{1}$$

where $\mathsf{E}[\cdot|\cdot]$ is the conditional expectation, $H(\cdot)$ is a strictly increasing link function and θ_0 is a vector of unknown parameters that satisfies $||\theta_0|| = 1$, where $||\cdot||$ denotes the L2-norm. The latter constraint is imposed in order to ensure the identifiability of the model. The explained Gini coefficient (Heuchenne and Jacquemain, 2022) is defined as

$$\mathrm{Gi}_{Y,X} = \mathrm{Gi}_{H(X^\intercal \theta_0)},$$

and satisfies $\mathrm{Gi}_{Y,X} \leq \mathrm{Gi}_Y$. It represents the inequality of Y that survives when, for each individual, the information contained in his or her covariates is used to predict Y. This definition is equivalent to

$$\mathrm{Gi}_{Y,X} = \max_\theta \frac{2\mathsf{COV}[Y, F_\theta(X^\intercal \theta)]}{\mathsf{E}[Y]}, \tag{2}$$

where $F_\theta(\cdot)$ is the CDF of $X^\intercal \theta$. In other words, the explained Gini coefficient is also the highest value that the concentration index of Y with respect to a linear combination of the covariates X can take. Interestingly, the formulation presented in (2) does not depend on $H(\cdot)$.

2.2 Estimation procedure and algorithm

The estimation procedure developed hereunder is based on Heuchenne and Jacquemain (2022), and on Jacquemain *et al.* (2022b) for the adaptation to sample weights. Let $(X_i^\intercal, Y_i)^\intercal$, with $i = 1, \ldots, n$ be an i.i.d sample with the same distribution as $(X^\intercal, Y)^\intercal$. Assume that we have at our disposal a vector of sample weights $\pi = (\pi_1, \ldots, \pi_n)^\intercal$ satisfying $\pi_i \geq 0 \ \forall i = 1, \ldots, n$ and $\sum_{i=1}^n \pi_i = 1$. We define the relative rank of observation i in the vector $X^\intercal\theta$ as

$$\hat{F}_i(\theta) = \sum_{j=1}^{n} \pi_j \bigg(\mathbb{1}\{X_i^\intercal\theta \geq X_j^\intercal\theta\} - \mathbb{1}\{X_i^\intercal\theta = X_j^\intercal\theta, U_i > U_j\} \bigg), \tag{3}$$

where $(U_1, \ldots, U_n)^\intercal$ are i.i.d uniform $(0, 1)$ random variables. The second indicator is an adaptation to ties and is discussed in Section 3 of Heuchenne and Jacquemain (2022). An alternative to this random allocation would consist in using the mean rank, where $\mathbb{1}\{X_i^\intercal\theta = X_j^\intercal\theta, U_i > U_j\}$ is replaced by $\frac{1}{2}\mathbb{1}\{X_i^\intercal\theta = X_j^\intercal\theta\}$.

In the Lorenz regression procedure, the vector θ_0 and the explained Gini coefficient are estimated with

$$\hat{\theta}_{\text{LR}} = \arg \max_{\theta, ||\theta||=1} \sum_{i=1}^{n} \pi_i Y_i \hat{F}_i(\theta) \tag{4}$$

$$\widehat{\text{Gi}}_{Y,X}^{\text{LR}} = \frac{2}{\overline{Y}} \sum_{i=1}^{n} \pi_i Y_i \hat{F}_i(\hat{\theta}_{\text{LR}}) - 1, \tag{5}$$

where $\overline{Y} = \sum_{i=1}^n \pi_i Y_i$. This estimation procedure does not entail to assume a specific form for $H(\cdot)$ nor does it need to be estimated. Notice however that $\theta \mapsto \hat{F}_i(\theta)$ is discrete. In the **LorenzRegression** package, the optimization programme displayed in (4) is solved via a genetic algorithm, implemented in the function `Lorenz.GA()`. Its usage is summarized as follows.

```
Lorenz.GA(YX_mat, ties.method=c("random","mean"),
          ties.Gini=c("random","mean"), seed.random=NULL, weights=NULL)
```

The only required argument `YX_mat` is a matrix of dimensions $n \times (p+1)$ where the first column is the response vector Y and the remaining ones are the covariates X. The arguments `ties.method` and `ties.Gini` indicate how ties are handled in the computation of $\hat{F}_i(\theta)$. By default, both arguments are set to `''random''`, meaning that Equation (3) is used. If `''mean''` is chosen, the mean rank solution is used instead. The existence of two separate arguments stems from the fact that ties influence first the estimation of θ_0 through Equation (4) and, second, the estimation of $\text{Gi}_{Y,X}$ through Equation (5). To ensure the comparability with other methods, one could choose to estimate the explained Gini coefficient with the mean rank solution, even though the random allocation is used to estimate θ_0. In that case, one would set `ties.method="random"` but `ties.Gini="mean"`. The argument `seed.random` imposes a fixed seed before

generation of the sample of uniform data when the random allocation is used. The default is `NULL`, in which case no seed is imposed. Finally, a vector of sample weights can be provided using the argument `weights`. By default, it is set to `NULL`, in which case all observations are given a weight of $1/n$. We refer the reader to the help file of `Lorenz.GA()` for details on additional arguments.

In a genetic algorithm, the idea is to start from a population of solution candidates and iteratively make this population evolve such that it is composed of solutions with an increasingly higher score. The evolution is governed by the principles of crossover and mutation, while the score is assessed via a fitness function. We refer the reader to Scrucca (2013) for a more complete description and the implementation in `R` via the **GA** package. Internally, `Lorenz.GA()` calls function `ga()` from the **GA** package, with the following fitness function

$$f(\theta) = \Big[1 - \Big|||\theta|| - 1\Big|\Big] \sum_{i=1}^{n} \pi_i Y_i \hat{F}_i(\theta),$$

where the factor multiplying the sum ensures that the final solution satisfies the norm constraint $||\theta|| = 1$. To speed up computations, the function is written in `C++` and integrated in `R` using **RcppArmadillo**.

3 The penalized Lorenz regression

In practice, an economist may want to determine the explained Gini coefficient from a large set of covariates, without having to perform a pre-selection prior to estimation. In that context, the Lorenz regression procedure detailed in the last section suffers from two issues. First, similarly to the R^2 in linear regression, $\widehat{\text{Gi}}^{\text{LR}}_{Y,X}$ never decreases and, in practice, slightly increases as we introduce new covariates, even if they are actually irrelevant. On a large set of covariates, the explained Gini coefficient is therefore prone to be overestimated. Second, the genetic algorithm may fail to converge when the number of covariates becomes large.

To solve these issues, Jacquemain *et al.* (2022a) introduce a penalized estimation procedure, where the discrete empirical distribution function displayed in (3) is replaced by a differentiable approximation. We now turn to this method.

3.1 Estimation procedure and algorithm

The penalized Lorenz regression solves

$$\hat{\theta}_{\text{PLR}} = \arg \max_{\theta, ||\theta||=1} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} \pi_i \pi_j Y_i K\left(\frac{X_i^\intercal \theta - X_j^\intercal \theta}{h}\right) - \sum_{k=1}^{p} p_\lambda(|\theta_k|) \right\}, \quad (6)$$

where $K(\cdot)$ is the integral of a kernel function, h is a bandwidth, $p_\lambda(\cdot)$ is a nonconcave penalty function and $\lambda > 0$ is a regularization parameter. We

advocate to use the SCAD penalty, which satisfies

$$p'_\lambda(x) = \begin{cases} \lambda & \text{if } x \leq \lambda \\ \frac{a\lambda - x}{a-1} & \text{if } \lambda < x \leq a\lambda \\ 0 & \text{if } x > a\lambda, \end{cases} \tag{7}$$

where $x > 0$ and $a > 2$ is an arbitrary constant. In what follows, we call "active" covariates those for which $\theta_{0,k} \neq 0$ and "non-active" those satisfying $\theta_{0,k} = 0$, where $\theta_{0,k}$ is the kth element of θ_0. The SCAD penalty ensures that the estimated coefficients of non-active covariates are set to 0 with a probability tending to one. Also, the vector of estimated weights associated to the active covariates is asymptotically normal, with a convergence rate unaffected by the selection process. We refer the reader to Theorem 2 in Jacquemain *et al.* (2022a) for more details. An alternative would consist in using the lasso penalty, where $p_\lambda(x) = \lambda x$. An estimator $\widehat{\text{Gi}}^{\text{PLR}}_{Y,X}$ for the explained Gini coefficient is obtained by plugging $\hat{\theta}_{\text{PLR}}$ in (5). In practice, inference can be performed using bootstrap.

In the **LorenzRegression** package, the penalized Lorenz regression may be implemented using either the lasso or the SCAD penalty. The former is solved using the FABS algorithm proposed by Shi *et al.* (2018), adapted to our objective function, and implemented in the function `Lorenz.FABS()`. The latter is solved using the SCAD-FABS algorithm proposed by Jacquemain *et al.* (2022a) and implemented in the function `Lorenz.SCADFABS()`.

In what follows, we focus on the `Lorenz.SCADFABS()` function. We first provide a brief explanation of the algorithm, then turn to the usage of the function. For further details, we refer the reader to Section 3.2 of Jacquemain *et al.* (2022a). The SCAD-FABS solves

$$\min_\theta Q(\theta) = L(\theta) + \sum_{k=1}^{p} p_\lambda(|\theta_k|)$$

where $Q(\cdot)$ is the objective function, $L(\cdot)$ is the loss function and $p_\lambda(\cdot)$ is the SCAD penalty. At each iteration, the algorithm updates only one coefficient, by a fixed amount of $+$ or $-$ ϵ, where $\epsilon > 0$ is the step size of the algorithm. This operation either reduces the size of the coefficient (backward step) or increases it (forward step). Let θ^t and θ^{t+1} denote the vectors of coefficients at iterations t and $t+1$. The update rule is then of the form

$$\begin{aligned} \theta^{t+1} &= \theta^t - \text{sign}(\theta^t_k)\mathbf{1}_k\epsilon && \text{(backward step)} \\ \theta^{t+1} &= \theta^t - \text{sign}\left(\nabla_k L(\theta^t)\right)\mathbf{1}_k\epsilon && \text{(forward step)} \end{aligned}$$

where $\mathbf{1}_k$ denotes the vector of size p taking value 1 at the kth component and 0 everywhere else, θ^t_k is the kth element of θ^t, ∇ is the gradient vector and ∇_k its kth element. The fact that a forward step necessarily increases the amplitude of the coefficient is settled by Lemma 1 in Jacquemain *et al.* (2022a). The index

k corresponding to that of the updated coefficient is determined by

$$k = \arg\min_{l \in \mathcal{A}^t} \left\{ -\nabla_l Q(\theta^t) \text{sign}(\theta_l^t) \right\} \qquad \text{(backward step)}$$
$$k = \arg\max_{l=1,\ldots,p} \left\{ |\nabla_l L(\theta^t)| - p'_\lambda(|\theta_l^t|) \right\} \qquad \text{(forward step)}$$

where $\mathcal{A}^t = \{k \in \{1, \ldots, p\} : \theta_k^t \neq 0\}$ and $p'_\lambda(|\theta_k^t|)$ is defined according to (7) for $|\theta_k^t| > 0$ and $p'_\lambda(|\theta_k^t|)$ is set to λ if $\theta_k^t = 0$. The SCAD-FABS solves the minimization problem for a path of λ values, which may be imposed by the user or determined within the algorithm, as in the FABS algorithm. In the penalized Lorenz regression, we incorporate the constraint $||\theta|| = 1$ by considering the Lagrangian function related to our problem. Accordingly, the loss function is

$$L(\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{n} \pi_i \pi_j Y_i K\left(\frac{X_i^\intercal \theta - X_j^\intercal \theta}{h}\right) + \gamma \sum_{k=1}^{p} \theta_k^2, \tag{8}$$

where $\gamma > 0$. The main arguments of the `Lorenz.SCADFABS()` function are described in the following code excerpt.

```
Lorenz.SCADFABS(YX_mat, weights=NULL, h, eps, a = 3.7,
                lambda="Shi", gamma = 0.05)
```

As in the function `Lorenz.GA()`, `YX_mat` and `weights` provide the data and optional sample weights. The remaining arguments correspond to tuning parameters of either the penalized problem or of the SCAD-FABS algorithm. The smoothness of the objective function is determined by the kernel and the bandwidth. Concerning the former, we impose $K(u) = [\frac{9}{8}u - \frac{5}{8}u^3 + \frac{1}{2}]\mathbb{1}\{-1 \leq u \leq 1\}$ to ensure good theoretical properties. The bandwidth h is determined via the argument `h`. The step size ϵ of the SCAD-FABS algorithm is set by the argument `eps`. The SCAD penalty depends on two parameters, a and λ. The former is ruled by the argument `a`, with default value of 3.7, as advised in Fan and Li (2001). The argument `lambda` takes three possible values. By default, it is set to `''Shi''`, where the path of λ values is determined within the algorithm, as in the FABS procedure of Shi *et al.* (2018). If set to `''grid''`, the path is defined by a grid, equidistant in the log scale. Otherwise, the user may provide a vector of values. Finally, the argument `gamma` corresponds to the Lagrange multiplier of Equation (8), with a default value of 0.05.

3.2 Choice for the bandwidth and regularization parameter

The smoothness of the kernel and the extent of penalization are crucial parameters of the estimation procedure. A careful choice for the bandwidth h and regularization parameter λ is therefore needed. The SCAD-FABS algorithm returns vectors of estimated coefficients for a path of λ values and a fixed h. As proposed in Jacquemain *et al.* (2022a), we advise to repeat this process over a

grid of values for the bandwidth satisfying $h = Cn^{-1/5.5}$. In the **LorenzRegression** package, three methods are available to select the pair (λ,h): BIC, bootstrap and cross-validation.

Denote by $\widehat{\text{Gi}}(\lambda, h)$ the estimated explained Gini coefficient obtained for given values of λ and h. A first option consists in choosing the pair (λ, h) that maximizes the following BIC criterion

$$\text{BIC-score}_{(\lambda,h)} = \log(\widehat{\text{Gi}}(\lambda, h)) - k_{(\lambda,h)} \frac{\log(n)}{2n},$$

where $k_{(\lambda,h)}$ is the number of covariates selected using (λ, h). This criterion is an adaptation of the BIC score proposed by Lin and Peng (2013). Another possibility benefits from the bootstrap procedure used to perform inference on the explained Gini coefficient. Let B denote the number of bootstrap resamples and $b = 1, \ldots, B$. At each iteration b, two samples are produced. First, a training sample (X_b^*, Y_b^*) is obtained by drawing with replacement from the original data. Second, a validation sample $(\tilde{X}_b^*, \tilde{Y}_b^*)$ is obtained as the out-of-bags, i.e. the data unused in the training sample. The SCAD-FABS algorithm is run on (X_b^*, Y_b^*) and yields an estimated vector of coefficients $\hat{\theta}_b^*(\lambda, h)$. An out-of-bag score for iteration b, $\widetilde{\text{Gi}}_b^*(\lambda, h)$, is obtained by plugging $\hat{\theta}_b^*(\lambda, h)$ and $(\tilde{X}_b^*, \tilde{Y}_b^*)$ in (5). The bootstrap selection procedure chooses the pair (λ, h) that maximizes the out-of-bag score

$$\text{OOB-score}_{(\lambda,h)} = \frac{1}{B} \sum_{b=1}^{B} \widetilde{\text{Gi}}_b^*(\lambda, h).$$

A final option uses K-fold cross-validation, where the data are split into K folds of roughly equivalent sizes. For each iteration, $(K-1)$ folds are used as training samples and the remaining one is used as validation sample. The procedure is then similar to the bootstrap selection. A cross-validation score is defined for each iteration, call it $\widetilde{\text{Gi}}_{-k}(\lambda, h)$. The cross-validation selection procedure chooses the pair (λ, h) that maximizes the cross-validation score

$$\text{CV-score}_{(\lambda,h)} = \frac{1}{K} \sum_{k=1}^{K} \widetilde{\text{Gi}}_{-k}(\lambda, h).$$

4 The LorenzRegression package

4.1 The `Gini.coef()`, `Lorenz.curve()` and `Lorenz.graphs()` functions

The **Lorenz.Regression** package offers several functions to analyze the inequality pattern characterizing some data. We first describe the function `Gini.coef()`, which computes a Gini coefficient or a concentration index. Its arguments are

- `y`: vector gathering the variable of interest for each observation.

- `x`: vector gathering the variable to use for the ranking of each observation. By default, it is set to `y` and the function computes the Gini coefficient of `y`. If the user supplies a vector `x`, the output becomes the concentration index of `y` with respect to `x`.
- `na.rm`: should missing values be deleted. Default value is `TRUE`. If `FALSE` is selected, missing values generate an error message.
- `ties.method`: determines how ties are handled. By default, the argument is set to `"mean"` and the average rank is used. If set to `random`, the random allocation is used instead.
- `seed`: fixes what seed is used if the random allocation is chosen. Default is `NULL`, in which case no seed is imposed.
- `weights`: vector of sample weights. Default is NULL, in which case each observation is given a weight of $1/n$, where n is the number of observations.

With similar arguments, the function `Lorenz.curve()` can either produce the Lorenz curve of `y` or the concentration curve of `y` with respect to `x`. The output is a function, whose unique argument is a proportion between 0 and 1. The function `Lorenz.curve()` admits an extra logical argument `graph`. If set to TRUE, a plot of the curve is displayed. The default value is FALSE, in which case no plot is produced.

Finally, function `Lorenz.graphs()` allows the user to plot simultaneously the Lorenz curve of a variable, as well as several concentration curves. It takes the following two main arguments:

- `formula`: standard formula object of the form `Y  X1 + X2 + ...`
- `data`: data frame containing the variables displayed in the formula.

The output of the function is a graph of the Lorenz curve of `Y` and of each of the concentration curves of `Y` with respect to the covariates `X1`, `X2`, ... indicated in the formula.

4.2 The `Lorenz.Reg()` function

This section discusses the core function of the **LorenzRegression** package. The function `Lorenz.Reg()` allows the user to fit a Lorenz or penalized Lorenz regression. Its usage is similar to other regression functions and several methods such as `print()`, `coef()`, `summary()`, `plot()` and `confint()` have been defined to provide detailed information on the fitted model. It uses the following arguments.

- `formula`: standard formula object of the form `Y  X1 + ...` determining the response variable and the covariates.
- `data`: data frame containing the variables displayed in the formula.

- `standardize`: whether covariates should be standardized prior to estimation. Default value is TRUE, in which case covariates are standardized. We advise not to set it to FALSE, especially if a penalized Lorenz regression is fitted. Without standardization, covariates are not treated on equal footing by the penalized procedure.
- `weights`: vector of sample weights. Default is NULL, in which case each observation is given a weight of $1/n$, where n is the number of observations.
- `parallel`: determines whether parallel computing should be used to distribute the computations between different CPUs. Default value is FALSE, in which case no parallel computing is performed. If set to TRUE, the number of core is set to `detectCores()`-1. The user can also supply a numerical value determining the number of cores to use.
- `penalty`: determines whether a non-penalized (`''none''`, default) or a penalized Lorenz regression should be fitted. In the latter case, the user has the choice between the SCAD (`''SCAD''`) and the lasso (`''LASSO''`) penalty.

The following arguments are particular to the penalized Lorenz regression.

- `h.grid`: grid of values for the bandwidth. The default is determined as

  ```
  h.grid=c(0.1,0.2,1,2,5)*nrow(data)^(-1/5.5)
  ```

- `eps`: step size of the algorithm. Default value is 0.005. Typically, lower values of `eps` lead to longer and finer paths but require more computation time.
- `sel.choice`: determines the selection method for the bandwidth and regularization parameter. As explained in Section 3, three methods are available: BIC, cross-validation and bootstrap. Possible values are any subset of the vector `c(''BIC'',''CV'',''Boot'')`. The default is `''BIC''`.

The next three parameters are only used if `sel.choice` contains `''CV''`.

- `nfolds`: number of folds. Default value is 10.
- `seed.CV`: value used as seed before the folds are formed. Default value is NULL, in which case no seed is imposed.
- `foldID`: determines how the folds are formed. The default value is NULL, in which case the folds are determined randomly. Otherwise, the user can supply a vector of size n determining the index of the fold for each observation.

Several parameters are particular to the bootstrap. In the case of a non-penalized Lorenz regression, bootstrap is only used to perform inference. If a penalized Lorenz regression is fitted instead, bootstrap is used both as selection method and to perform inference.

- `Boot.inference`: determines whether bootstrap inference should be produced. Since bootstrap may require a lot of computation time, the default value is FALSE. In the case of a penalized Lorenz regression, this parameter is automatically turned to TRUE if `sel.choice` contains `''Boot''`. Conversely, if `Boot.inference` is set to TRUE, `''Boot''` is added to `sel.choice`.
- `B`: number of bootstrap repetitions. Default value is 500.
- `alpha`: significance level for the bootstrap confidence intervals. Default is 0.05.
- `bootID`: determines how the bootstrap resamples are formed. The default value is NULL, in which case the samples are drawn with replacement from the original data. Otherwise, the user can supply a matrix where each row provides the ID of the observations selected in each bootstrap resample.
- `seed.boot`: value used as seed before the bootstrap resamples are formed. Default value is NULL, in which case no seed is imposed.

The last two arguments are mainly useful when the estimation is performed in several steps. For example, the original estimation is fitted on a single computer and the bootstrap iterations are distributed between different machines.

- `LR`: output of a call to `Lorenz.GA()` or to `PLR.wrap()`. The latter is a wrapper of functions `Lorenz.FABS()` and `Lorenz.SCADFABS()`. We refer the reader to the help of this function for extra information.
- `LR.boot`: output of a call to `Lorenz.boot()`, which allows the user to perform bootstrap for the Lorenz and penalized Lorenz regressions. Again we refer the reader to the help for further details.

Finally, additional parameters corresponding to arguments of functions `Lorenz.GA()`, `Lorenz.FABS()` and `Lorenz.SCADFABS()` can be passed in the `...` argument.

If the `penalty` argument is set to `none`, the function outputs an object of class `LR`. Otherwise, it outputs an object of class `PLR`. The associated S3 methods are illustrated in the next section.

5 Illustration

We start by loading the packages required in this example.

```
R> library(LorenzRegression)
R> library(ineq)
```

Throughout this illustration, we use the dataset `Ilocos` from the `R` package **ineq** (Zeileis, 2014), which contains economic data on 632 households in the Philippines. We use total household income (`income`) as response variable and the following covariates :

- `sex`: a factor with two levels, `"male"` (518 observations) and `"female"` (114 observations), indicating the biological sex of the household head;
- `family.size`: a numerical variable indicating the family size. Results range from 1 to 13, with a mean of 5.19;
- `urbanity`: a factor with two levels, `"rural"` (301 observations) and `"urban"` (331 observations);
- `province`: a factor with four levels: `"Ilocos Norte"` (65 observations), `"Ilocos Sur"` (68 observations), `"La Union"` (116 observations) and `"Pangasinan"` (383 observations).

The data are loaded with

```
R> data("Ilocos", package="ineq")
```

Descriptive analysis

The Gini coefficient of `income` and the concentration index of `income` with respect to `family.size` are computed using `Gini.coef()`.

```
R> Gi <- Gini.coef(Ilocos$income)
R> Gi
[1] 0.43
R> CI.mean <- Gini.coef(Ilocos$income,Ilocos$family.size)
R> CI.mean
[1] 0.089
```

Since `family.size` is a discrete covariate, ties appear in the ordering generated by this variable. In the computation of the concentration index, the mean rank is used by default. In what follows, we use the random allocation instead, repeating the operation 1000 times with a different seed.

```
R> CI.random <- sapply(1:1000, function(i)Gini.coef(Ilocos$income,Ilocos$family.size,
+                                                   ties.method = "random", seed = i))
R> boxplot(CI.random)
R> abline(h=CI.mean)
```

Figure 1 displays a boxplot of the concentration indices thus obtained. The distribution is centered on the solution obtained with the mean rank, indicated by the horizontal line. The size of the boxplot indicates how the concentration index varies as we change the rule determining how ties are broken. We now compute the Lorenz curve of `income` and evaluate it on a small grid of values.

```
R> LC <- Lorenz.curve(Ilocos$income)
R> p <- seq(0,1,length.out=5)
R> LC(p)
[1] 0.000 0.079 0.214 0.446 1.000
```

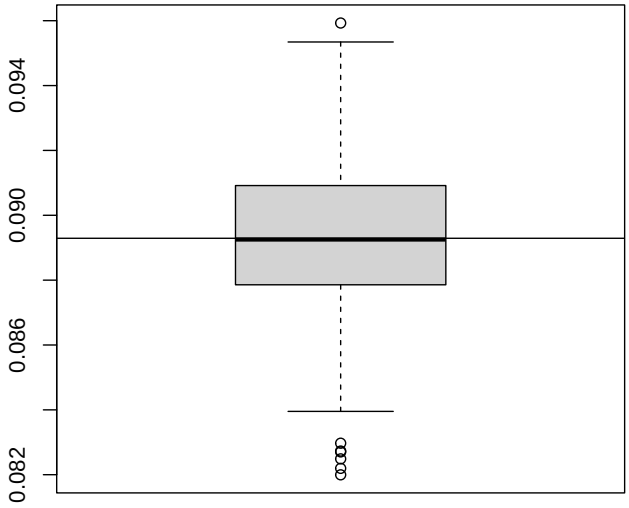

Figure 1: Concentration index of `income` with respect to `family.size` - Random allocation

Finally, we display the Lorenz curve of `income` and the concentration curve of `income` with respect to `family.size` in Figure 2. They are obtained using the function `Lorenz.graphs()`.

```
R> Lorenz.graphs(income ~ family.size, data = Ilocos)
```

The Lorenz regression

We fit the Lorenz regression to the `Ilocos` data, with `income` as response variable. The covariates include `sex`, `family.size`, `urbanity` and `province`. We also include interactions between `sex` and `family.size`, and between `sex` and `urbanity`. We use the mean rank in the estimation of the explained Gini coefficient, by setting `ties.Gini = "mean"`.

```
R> LR <- Lorenz.Reg(income ~ sex*family.size + sex*urbanity + province,
+                   data = Ilocos, ties.Gini = "mean", seed.random = 456,
+                   Boot.inference = TRUE)
```

The vector of estimated coefficients is retrieved with method `coef()`.

```
R> coef(LR)
              sexmale           family.size         urbanityurban
                0.318                 0.126                 0.845
   provinceIlocos Sur      provinceLa Union    provincePangasinan
                0.071                -0.219                -0.226
```

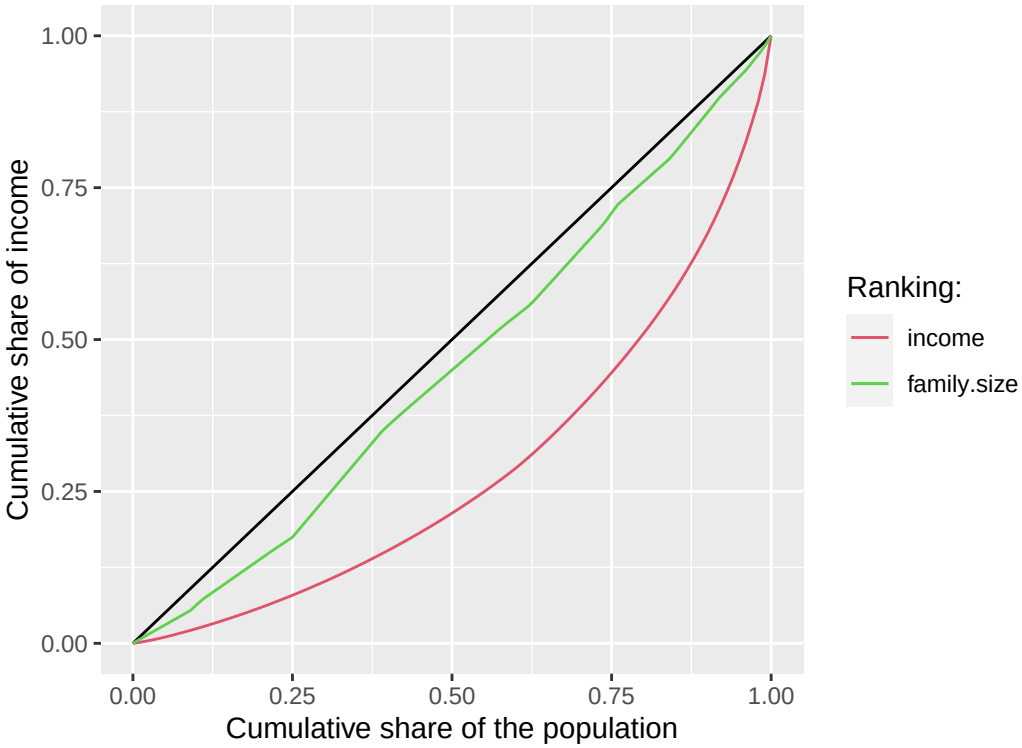

Figure 2: Lorenz curve of `income` and concentration curve of `income` with respect to `family.size`

```
sexmale:family.size sexmale:urbanityurban
             -0.028                -0.254
```

A summary of the model is provided with method `summary()`. The estimated explained Gini coefficient is of 15.4%, which corresponds to 36.2% of the observed inequality. The summary also displays a table with the estimated coefficients. Let us focus on covariate `family.size`. In Figure 2, we observed a positive association between `income` and `family.size`. This is confirmed by the regression model since the estimated coefficient is positive. In the descriptive analysis, we also computed a concentration index of 8.9%. This would be the value of the explained Gini coefficient if we included only `family.size` in the regression. Including the remaining covariates helps us rise explained inequality from 8.9% to 15.5%.

```
R> summary(LR)
The explained Gini coefficient is of 0.15452

The Lorenz-R2 is of 0.36191
```

```
Estimated coefficients and associated p-values
|                       | estimate| p-value|
|:----------------------|--------:|-------:|
|sexmale                |     0.32|    0.21|
|family.size            |     0.13|    0.01|
|urbanityurban          |     0.84|    0.00|
|provinceIlocos Sur     |     0.07|    0.75|
|provinceLa Union       |    -0.22|    0.20|
|provincePangasinan     |    -0.23|    0.12|
|sexmale:family.size    |    -0.03|    0.43|
|sexmale:urbanityurban  |    -0.25|    0.33|
```

Because we set the argument `Boot.inference` to `TRUE`, p-values were computed for each coefficient. At a 5%-level of significance, we see that only `family.size` and `urbanityurban` are significant. A confidence interval for the explained Gini coefficient is obtained with method `confint()`.

```
R> confint(LR)
Lower bound Upper bound
       0.12        0.19
```

By default, 95% confidence intervals are constructed, but the level can be changed using the argument `level`. Three bootstrap methods are available and are chosen via the argument `boot.method`. Parametric boostrap (`"Param"`, the default) uses the asymptotic normality of the estimated explained Gini coefficient and estimates the variance by bootstrap. Percentile bootstrap (`"Perc"`) directly plugs in the quantiles of the estimated explained Gini coefficient obtained on the bootstrap resamples. Finally, basic bootstrap (`"Basic"`) is based on bootstrapping the whole distribution of the estimator.

Confidence intervals for each individual coefficient can be obtained by changing the `parm` argument to `''theta''`. We also change the significance level and the bootstrap method to illustrate the use of arguments `level` and `boot.method`.

```
R> confint(LR, parm = "theta", level = 0.9, boot.method = "Perc")
                      Lower bound Upper bound
sexmale                    -0.373       0.480
family.size                 0.022       0.175
urbanityurban               0.601       0.918
provinceIlocos Sur         -0.245       0.470
provinceLa Union           -0.375       0.179
provincePangasinan         -0.399       0.106
sexmale:family.size        -0.056       0.059
sexmale:urbanityurban      -0.520       0.343
```

We can also display explained inequality via concentration and Lorenz curves.

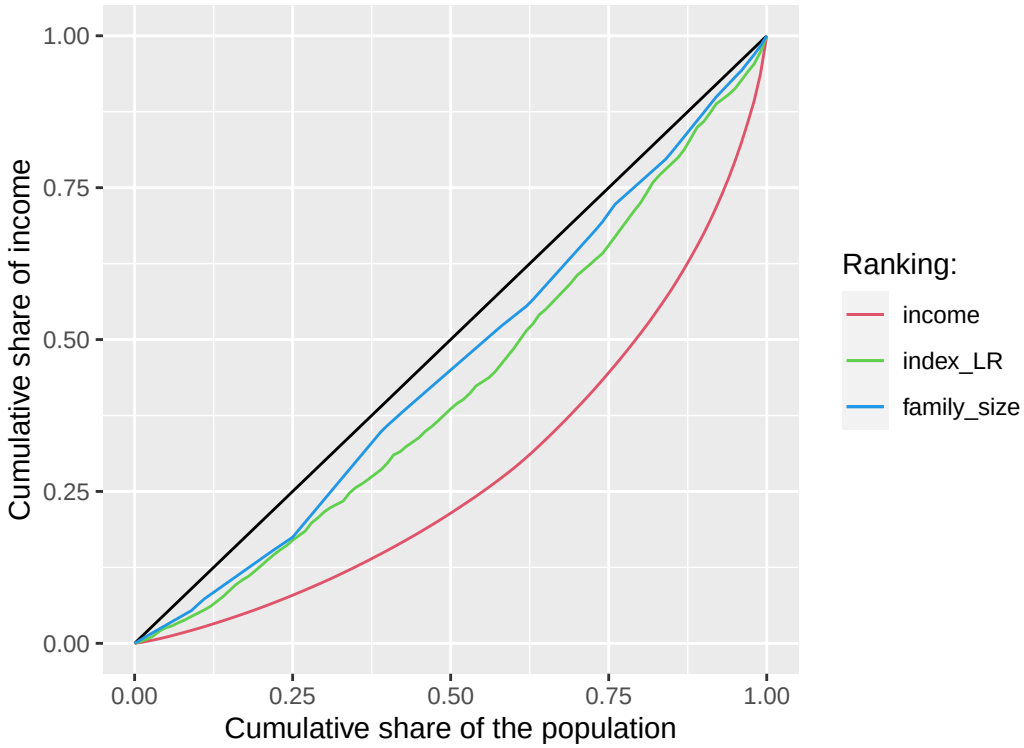

Figure 3: Lorenz curve of `income`, concentration curve of `income` with respect to `family.size` and concentration curve of `income` with respect to the estimated index

In the following piece of code, we make use of the function `Lorenz.graphs()` to produce Figure 3. The red curve is the Lorenz curve of `income`. The blue curve is the concentration curve of `income` with respect to `family.size`. Finally, the green curve is the concentration curve of `income` with respect to the index estimated by the Lorenz regression. The closer the green curve to the red curve, the greater the Lorenz-R^2, i.e. the more observed inequality is explained by the regression model.

```
R> data.plot <- cbind(LR$Fit,Ilocos$family.size)
R> colnames(data.plot) <- c("income","index_LR","family_size")
R> Lorenz.graphs(income ~ ., data.plot)
```

The penalized Lorenz regression

We fit the penalized Lorenz regression to the `Ilocos` data, with the same re-

sponse variable and covariates as before. We use the SCAD penalty and choose the couple (h, λ) by BIC and bootstrap. The step size of the SCAD-FABS algorithm is set to $\epsilon = 0.01$. Finally, parallel computing is used to distribute the bootstrap iterations across the available CPUs.

```
R> PLR <- Lorenz.Reg(income ~ sex*family.size + sex*urbanity + province,
+                     data = Ilocos,
+                     penalty = "SCAD",
+                     sel.choice = c("BIC","Boot"),
+                     seed.boot = 123,
+                     eps = 0.01,
+                     parallel = TRUE)
```

The coefficients estimated by the SCAD-FABS algorithm are obtained with method `coef()`. The output is a list where each element corresponds to a method of selection for the couple (λ, h).

```
R> coef(PLR, renormalize = FALSE)
$BIC
                sexmale             family.size           urbanityurban
                  0.000                   0.119                   0.767
   provinceIlocos Norte      provinceIlocos Sur        provinceLa Union
                  0.118                   0.247                   0.000
     provincePangasinan   sexfemale:family.size     sexmale:family.size
                 -0.178                   0.023                   0.000
sexfemale:urbanityrural   sexmale:urbanityrural sexfemale:urbanityurban
                 -0.539                   0.000                   0.000
  sexmale:urbanityurban
                  0.000

$Boot
                sexmale             family.size           urbanityurban
                  0.000                   0.099                   0.995
   provinceIlocos Norte      provinceIlocos Sur        provinceLa Union
                  0.000                   0.000                   0.000
     provincePangasinan   sexfemale:family.size     sexmale:family.size
                  0.000                   0.000                   0.000
sexfemale:urbanityrural   sexmale:urbanityrural sexfemale:urbanityurban
                  0.000                   0.000                   0.000
  sexmale:urbanityurban
                  0.000
```

In presence of categorical covariates, the fit of the penalized Lorenz regression would depend on the chosen reference category, see Jacquemain *et al.* (2022b). We avoid this issue by introducing all dummies in the regression, except for variables with only two categories. This is the reason why the four provinces are included in the output of `coef(PLR, renormalize = FALSE)`. Once the estimation is performed, the vector of coefficients can be transformed to match a specific choice of reference categories. By setting `renormalize=TRUE` (the default), the reference categories are chosen as in the non-penalized case, i.e. as the first categories in the alphabetical order. Method `summary()` displays these transformed coefficients as well as information about the model fit.

```
R> summary(PLR)
Summary of the model fit
|     | Explained Gini| Lorenz-R2| lambda|     h| Number of variables| BIC score| Boot score|
|:----|--------------:|---------:|------:|-----:|-------------------:|---------:|----------:|
|BIC  |         0.1541|    0.3609|  114.1| 1.548|                   7|    -1.906|     0.1306|
|Boot |         0.1435|    0.3362| 3442.9| 1.548|                   2|    -1.951|     0.1346|

Estimated coefficients
|                       |     BIC|   Boot|
|:----------------------|-------:|------:|
|sexmale                |  0.3462| 0.0000|
|family.size            |  0.0910| 0.0989|
|urbanityurban          |  0.8386| 0.9951|
|provinceIlocos Sur     |  0.0832| 0.0000|
|provinceLa Union       | -0.0756| 0.0000|
|provincePangasinan     | -0.1897| 0.0000|
|sexmale:family.size    | -0.0147| 0.0000|
|sexmale:urbanityurban | -0.3462| 0.0000|
```

The summary of the model fit includes the estimated explained Gini coefficient and Lorenz-R^2, the selected couple (λ, h), the number of included variables, as well as the BIC and bootstrap scores. Both selection methods lead to the same choice for the bandwidth, i.e. $h = 1.548$. The bootstrap selection method favours a larger penalty: $\lambda = 3273.96$, against $\lambda = 114.13$ for the BIC criterion, and therefore yields a sparser model. The amount of sparsity is represented by the number of non-zero coefficients before choosing a reference category, which are displayed in column `Number of variables`. With the bootstrap selection, only 2 variables have a non-zero coefficients, against 7 variables with the BIC procedure. The latter yields an output very similar to the non-penalized regression. The estimated Gini coefficient is of 15.41%, against 15.45% in the non-penalized case. The estimated coefficients are also of similar magnitudes. The bootstrap procedure yields an estimated Gini coefficient of 14.38%.

```
R> plot(PLR)
```

Method `plot()` displays several graphs. The first plot is the Lorenz curve of the response, along with the concentration curves of the response with respect to the index obtained by each selection method. We do not display this graph here because we want to compare with the fit obtained with the non-penalized Lorenz regression. Therefore, we display instead Figure 4, obtained by running the following piece of code.

```
R> data.plot$index_PLR.BIC <- PLR$Fit$Index.BIC
R> data.plot$index_PLR.Boot <- PLR$Fit$Index.Boot
R> Lorenz.graphs(income ~ ., data.plot)
```

As expected, the concentration curve obtained with BIC is very close to that obtained with the non-penalized procedure. The concentration curve obtained with bootstrap reproduces less inequality, since it lies closer to the 45° line. The second plot, given in Figure 5, displays the evolution of the estimated coefficients (before transformation related to the choice of reference categories)

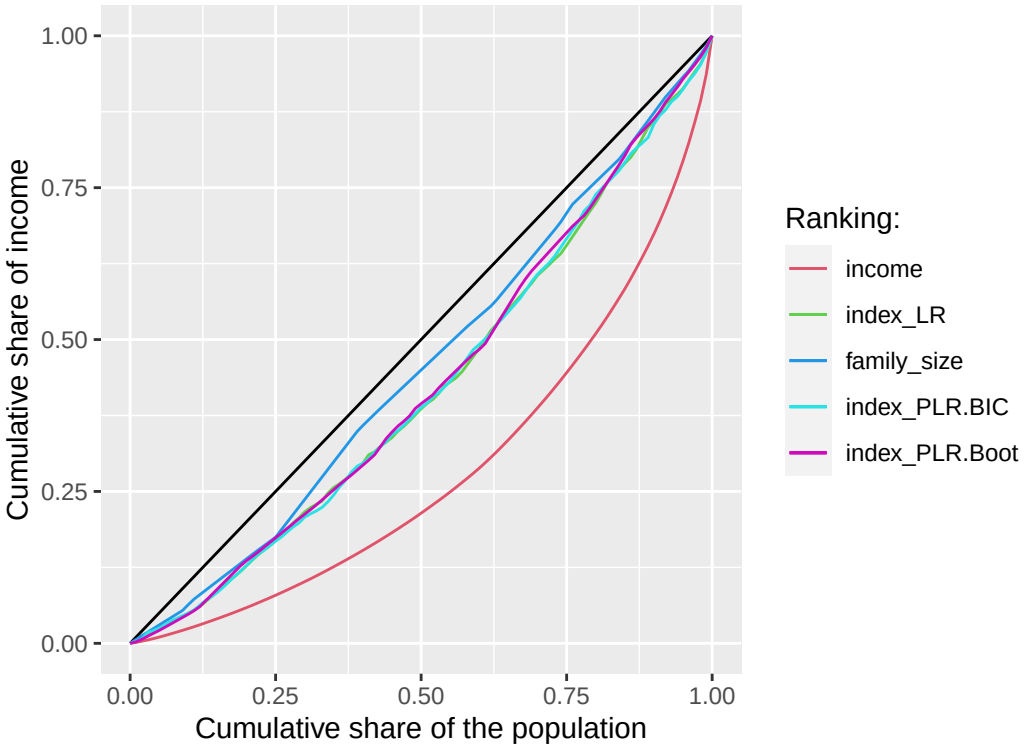

Figure 4: Lorenz curve of `income`, concentration curve of `income` with respect to `family.size` and concentration curves of `income` with respect to each estimated index

as the penalty relaxes. At the beginning, the penalty is the largest and only one coefficient is included. This coefficient corresponds to `urbanityurban` and it takes a value of 1 because of the norm constraint. As the algorithm proceeds, the penalty decreases and new variables enter the model. For example, `family.size` is the second variable to enter. The third plot, provided in Figure 6, shows the evolution along the path of the scores corresponding to each selection method. To ease the comparison, the scores are normalized so that the optimum is attained at 1. The bootstrap score yields a well-defined maximum in the beginning of the path, producing a highly sparse model. The BIC score, however, is relatively flat along the path. Therefore, we would favour using the former. Figures 5 and 6 indicate that the bandwidth was selected by BIC. Since the whole path depends on the choice of h, these two plots are reproduced for each selection method. In this illustration, the same bandwidth is selected with both methods. The plots related to the bootstrap are exactly the same as those obtained for BIC and are therefore not reproduced.

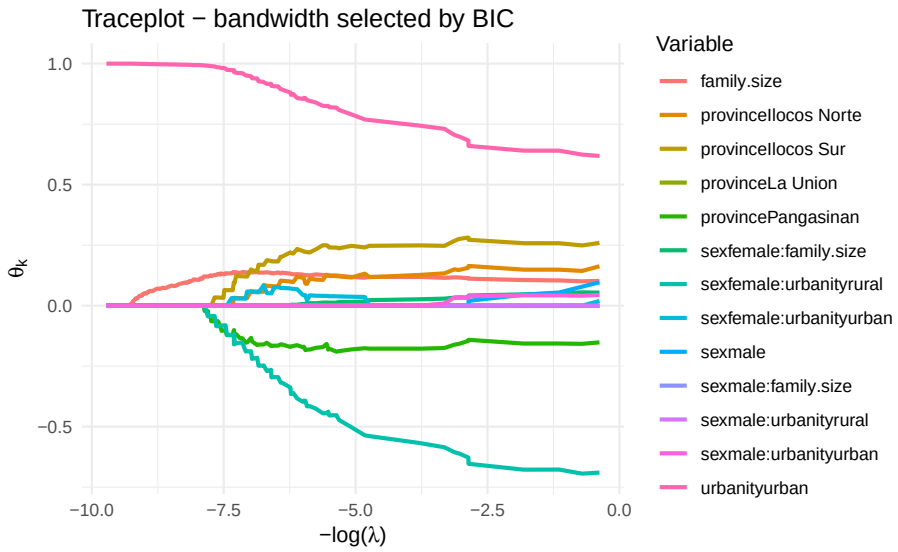

Figure 5: Evolution of the estimated coefficients as the penalty relaxes

Finally, we obtain confidence intervals for the explained Gini coefficient using the method `confint()`.

```
R> confint(PLR)
     Lower bound Upper bound
BIC         0.12        0.19
Boot        0.10        0.18
```

6 Summary and discussion

In this paper, we demonstrate the use of the **LorenzRegression** package, which implements the non-penalized and penalized Lorenz regressions. These methods are appropriate whenever one is interested in explaining the inequality in a response variable with a set of covariates. They provide inference for the explained Gini coefficient, a measure of explained inequality assuming a single-index model. The penalized procedure allows for an automatic selection of the relevant covariates.

The illustration developed in Section 5 shows that the fitting of the model is similar to standard regression techniques. It is performed with the function `Lorenz.Reg()` and it benefits from several S3 methods. Information on the model fit is obtained numerically with `summary()`, or graphically with `plot()`. Confidence intervals for the explained Gini coefficient are obtained with `confint()`. As a side advantage, the package allows the computation and graphical representation of Lorenz and concentration curves, as well as the com-

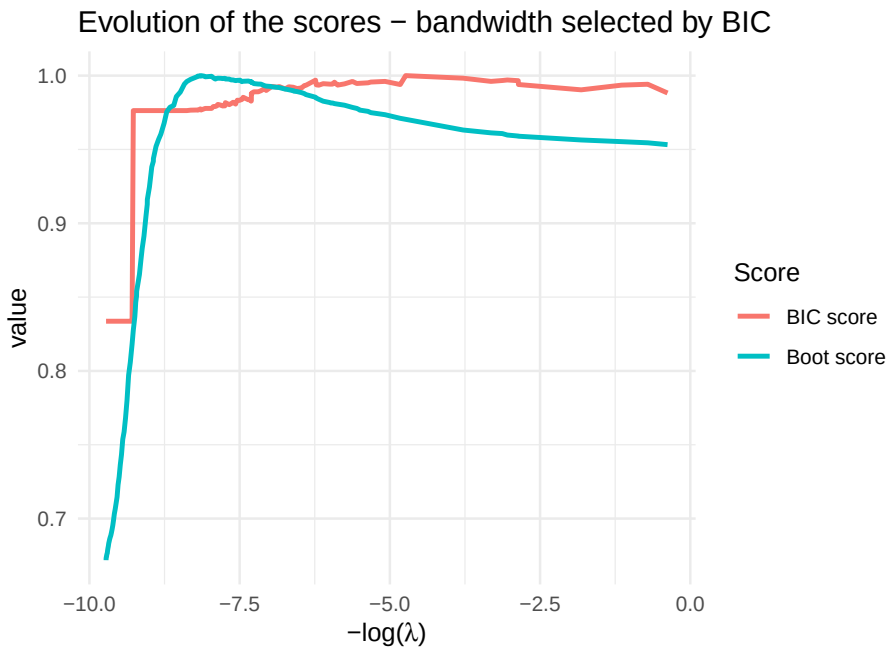

Figure 6: Evolution of the scores for each selection method as the penalty relaxes

putation of a Gini coefficient or a concentration index. All functions allow the user to specify sample weights.

Computational details

Computations performed in this paper use `R` 4.1.2 and the **LorenzRegression** 1.0.0 package. Both can be obtained from the Comprehensive `R` Archive Network (CRAN) at `https://CRAN.R-project.org/`.

References

Alfons A, Templ M (2012). "Estimation of Social Exclusion Indicators from Complex Surveys: The R Package Laeken." doi:10.2139/ssrn.2244876.

Devleesschauwer B, Willimes S, Van Malderen C, Konings P, Speybroeck N (2017). *rineq: Statistical Analysis of Health Inequalities*. R package version 0.0.1, URL `https://github.com/brechtdv/rineq/`.

Eddelbuettel D, Balamuta JJ (2018). "Extending R with C++: A Brief Intro-

duction to Rcpp." *The American Statistician*, **72**(1), 28–36. ISSN 0003-1305. doi:10.1080/00031305.2017.1375990.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo." *Computational Statistics & Data Analysis*, **71**(C), 1054–1063. ISSN 0167-9473. doi:10.1016/j.csda.2013.02.005.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360. ISSN 0162-1459. doi:10.1198/016214501753382273.

Heuchenne C, Jacquemain A (2022). "Inference for Monotone Single-Index Conditional Means: A Lorenz Regression Approach." *Computational Statistics & Data Analysis*, **167**(C). ISSN 0167-9473. doi:10.1016/j.csda.2021.107347.

Jacquemain A, Heuchenne C, Pircalabelu E (2022a). "A Penalised Bootstrap Estimation Procedure for the Explained Gini Coefficient."

Jacquemain A, Heuchenne C, van Kerm P (2022b). "A Penalized Lorenz Regression Analysis of Inequality of Opportunity."

Lin H, Peng H (2013). "Smoothed Rank Correlation of the Linear Transformation Regression Model." *Computational Statistics & Data Analysis*, **57**, 615–630. doi:10.1016/j.csda.2012.07.012.

Machowska K, Napora J, Wójcik S (2022). *wINEQ: Inequality Measures for Weighted Data*. R package version 1.0.1, URL `https://CRAN.R-project.org/package=wINEQ`.

Microsoft, Weston S (2022). *foreach: Provides Foreach Looping Construct*. R package version 1.5.2, URL `https://CRAN.R-project.org/package=foreach`.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Scrucca L (2013). "GA: A Package for Genetic Algorithms in R." *Journal of Statistical Software*, **53**, 1–37. ISSN 1548-7660. doi:10.18637/jss.v053.i04.

Shi X, Huang Y, Huang J, Ma S (2018). "A Forward and Backward Stagewise Algorithm for Nonconvex Loss Functions with Adaptive Lasso." *Computational Statistics & Data Analysis*, **124**, 235–251. ISSN 0167-9473. doi:10.1016/j.csda.2018.03.006.

Speybroeck N, Konings P, Lynch J, Harper S, Berkvens D, Lorant V, Geckova A, Hosseinpoor AR (2010). "Decomposing Socioeconomic Health Inequalities." *International Journal of Public Health*, **55**(4), 347–351. ISSN 1661-8564. doi:10.1007/s00038-009-0105-z.

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL `http://ggplot2.org`.

Zeileis A (2014). *ineq: Measuring Inequality, Concentration, and Poverty*. R package version 0.2-13, URL `https://CRAN.R-project.org/package=ineq`.