Estimating Linear Models in R:

from Ordinary Least Squares to Maximum Entropy via GCEstim

</> CODE

AUTHOR
Jorge Cabral

AFFILIATIONS
Department of Mathematics, University of Aveiro, Aveiro, Portugal
CIDMA, University of Aveiro, Aveiro, Portugal

PUBLISHED
November 5, 2025

1 Introduction

One of the most widely used data analysis techniques is univariate multiple linear regression. In general, it proposes to mathematically model the relation between a dependent variable and a set of independent variables (or predictors) in order to understand whether the latter predict and/or explain the former. In the modelling process some assumptions should be imposed (Greene 2008).

Data transformations are sometimes applied, being one of those the standardization of the variables. This is done by subtracting the mean from each and then dividing by their respective standard deviation. The estimated coefficients (from now on referred as coefficients) are called standardized coefficients (Bring 1994).

Consider the univariate multiple linear regression model given by

$$y = X\beta + \epsilon,$$

where y denotes a $(N \times 1)$ vector of noisy observations, $N \in \mathbb{N}$, β is a $((K+1) \times 1)$ vector of unknown parameters or coefficients, $K \in \mathbb{N}$, X is a known $(N \times (K+1))$ design matrix and ϵ is a $(N \times 1)$ vector of random disturbances (errors), usually assumed to have a conditional expected value of zero and representing spherical disturbances, i.e, $E\left[\epsilon|X\right] = 0$ and $E[\epsilon\epsilon'|X] = \sigma^2 I$, where I is a $(N \times N)$ identity matrix and σ^2 is the error variance.

The univariate multiple linear regression model can be written as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_K x_K + \epsilon,$$

and standardizing y and $x_j, j \in \{1, \ldots, K\}$ the following results are obtained

$$\begin{aligned} y^* &= X^* b + \epsilon^* \\ y^* &= b_0 + b_1 x_1^* + b_2 x_2^* + \cdots + b_K x_K^* + \epsilon^*, \end{aligned}$$

where

$$\begin{aligned} y_i^* &= \frac{y_i - \frac{\sum_{i=1}^N y_i}{N}}{\sqrt{\frac{1}{N}\sum_{i=1}^N \left(y_i - \frac{\sum_{i=1}^N y_i}{N}\right)^2}}, \\ x_{ji}^* &= \frac{x_{ji} - \frac{\sum_{i=1}^N x_{ji}}{N}}{\sqrt{\frac{1}{N}\sum_{i=1}^N \left(x_{ji} - \frac{\sum_{i=1}^N x_{ji}}{N}\right)^2}}, \\ b_j &= \frac{\sqrt{\frac{1}{N}\sum_{i=1}^N \left(x_{ji} - \frac{\sum_{i=1}^N x_{ji}}{N}\right)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^N \left(y_i - \frac{\sum_{i=1}^N y_i}{N}\right)^2}} \beta_j, \end{aligned}$$

with $j \in \{1, \ldots, K\}$, and $b_0 = 0$. In this formulation, b_j are called standardized coefficients.

1.1 Ordinary Least Squares

The Ordinary Least Squares (OLS) estimator of β takes the form

$$\begin{aligned} \widehat{\beta}^{OLS} = \begin{bmatrix} \widehat{\beta}_0^{OLS} \\ \widehat{\beta}_1^{OLS} \\ \widehat{\beta}_2^{OLS} \\ \vdots \\ \widehat{\beta}_K^{OLS} \end{bmatrix} &= \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2 \\ &= \left(X'X\right)^{-1} X'y, \end{aligned}$$

where X' is the transpose of X.

1.2 Ridge regression

The Ridge regression introduced by Hoerl and Kennard (Hoerl and Kennard 1970) is an estimation procedure to handle collinearity without removing variables from the regression model. By adding a small non-negative constant (ridge or shrinkage parameter) to the diagonal of the correlation matrix of the explanatory variables, it is possible to reduce the variance of the OLS estimator through the introduction of some bias. Although the resulting estimators are biased, the biases are small enough for these estimators to be substantially more precise than the unbiased estimators. The challenge in Ridge regression remains on the selection of the ridge parameter. One straightforward approach is based on simply plotting the coefficients against several possible values for the ridge parameter and inspecting the resulting traces. The Ridge Regression estimator of λ takes the form

$$\begin{aligned}\widehat{\beta}^{ridge} &= \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2 + \lambda\|\beta\|^2 \\ &= (X'X + \lambda I)^{-1}X'y,\end{aligned}$$

where $\lambda \geq 0$ denotes the ridge parameter and I is a $(K \times K)$ identity matrix. Note that when $\lambda \to 0$, the Ridge regression estimator approaches the OLS estimator whereas the Ridge regression estimator approaches the zero vector when $\lambda \to \infty$. Thus, a trade-off between variance and bias is needed.

1.3 Generalized Maximum Entropy estimator

Golan et al (Golan, Judge, and Miller 1996) generalized the Maximum Entropy formalism (Jaynes 1957) to linear inverse problems with noise, expressed in the previous chapter linear model. The idea is to treat each β_j, $j \in \{0, \ldots, K\}$, as a discrete random variable with a compact support and $2 \leq M < \infty$ possible outcomes, and each ϵ_i, $i \in \{1, \ldots, N\}$, as a finite and discrete random variable with $2 \leq J < \infty$ possible outcomes. Assuming that both the unknown parameters and the unknown error terms may be bounded a priori, the linear model can be presented as

$$Y = XZp + Vw,$$

where

$$\beta = Zp = \begin{bmatrix} z_1' & 0 & \cdots & 0 \\ 0 & z_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z_K' \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_K \end{bmatrix},$$

with Z a $(K \times (K \times M))$ matrix of support values and p a $((K \times M) \times 1)$ vector of unknown

probabilities, and

$$\epsilon = Vw = \begin{bmatrix} v_1' & 0 & \cdots & 0 \\ 0 & v_2' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_N' \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix},$$

with V a $(N \times (N \times J))$ matrix of support values and w a $((N \times J) \times 1)$ vector of unknown probabilities. For the linear regression model, the Generalized Maximum Entropy (GME) estimator is given by

$$\hat{\beta}^{GME}(Z, V) = \underset{p,w}{\operatorname{argmax}} \left\{ -p' \ln p - w' \ln w \right\},$$

subject to the model constraint

$$Y = XZp + Vw,$$

and the additivity constraints for p and w, respectively,

$$1_K = (I_K \otimes 1_M')p,$$
$$1_N = (I_N \otimes 1_J')w,$$

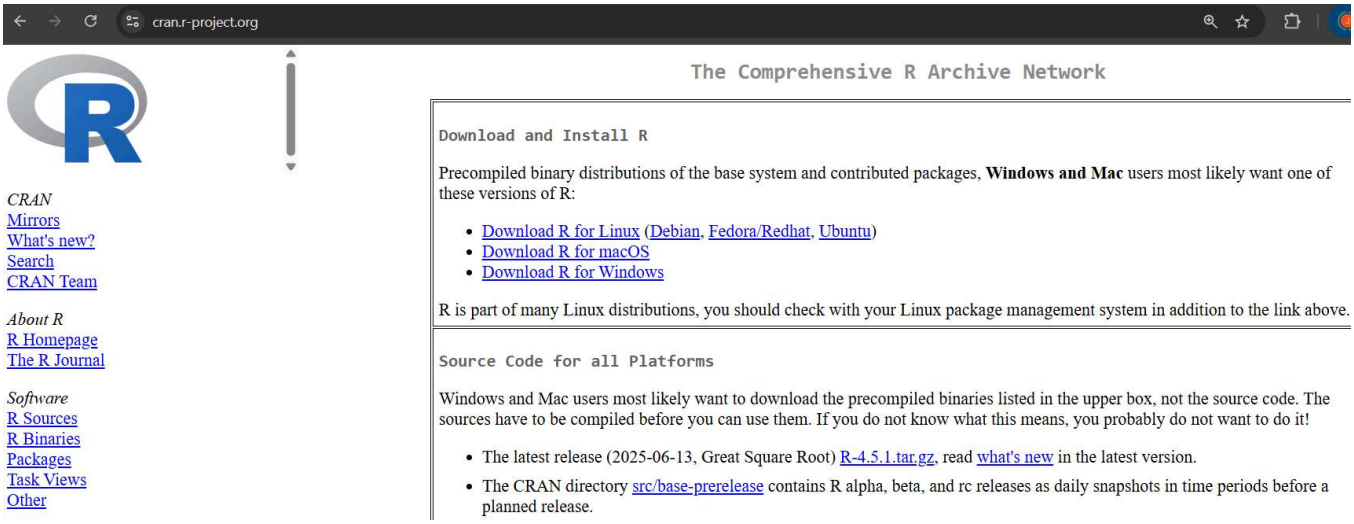
where $\otimes$ represents the Kronecker product, 1 is a column vector of ones with a specific dimension, I is an identity matrix with a specific dimension and Z and V are the matrices of supports, and $p > 0$ and $w > 0$ are probability vectors to be estimated. The GME estimator generates the optimal probability vectors $\widehat{p}$ and $\widehat{w}$ that can be used to form point estimates of the unknown parameters and the unknown random errors. Since the objective function is strictly concave in the interior of the additivity constraint set, a unique solution for the GME estimator is guaranteed if the intersection of the model and the additivity constraint sets is non-empty (Golan, Judge, and Miller 1996). The supports in matrices Z and V are defined as being closed and bounded intervals within which each parameter or error is restricted to lie, implying that researchers need to provide exogenous information (which, unfortunately, it is not always available). This is considered the main weakness of the GME estimator (Caputo and Paris 2008). Golan et al (Golan, Judge, and Miller 1996) discuss these issues in the case of minimal prior information: for the unknown parameters, the authors recommend the use of wide bounds (this is naturally subjective) for the supports in Z, without extreme risk consequences; for the unknown errors, the authors suggest the use of the three-sigma rule with a sample scale parameter (Pukelsheim 1994). The number of points M in the supports is less controversial and are usually used in the literature between 3 and 7 points, since there is likely no significant improvement in the estimation with more points in the supports. The three-sigma rule, considering the standard deviation of the noisy observations and $J = 3$ points in the supports is usually adopted.

2 Setting up

2.1 R software

> "R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS." — R software (RCoreTeam 2025)

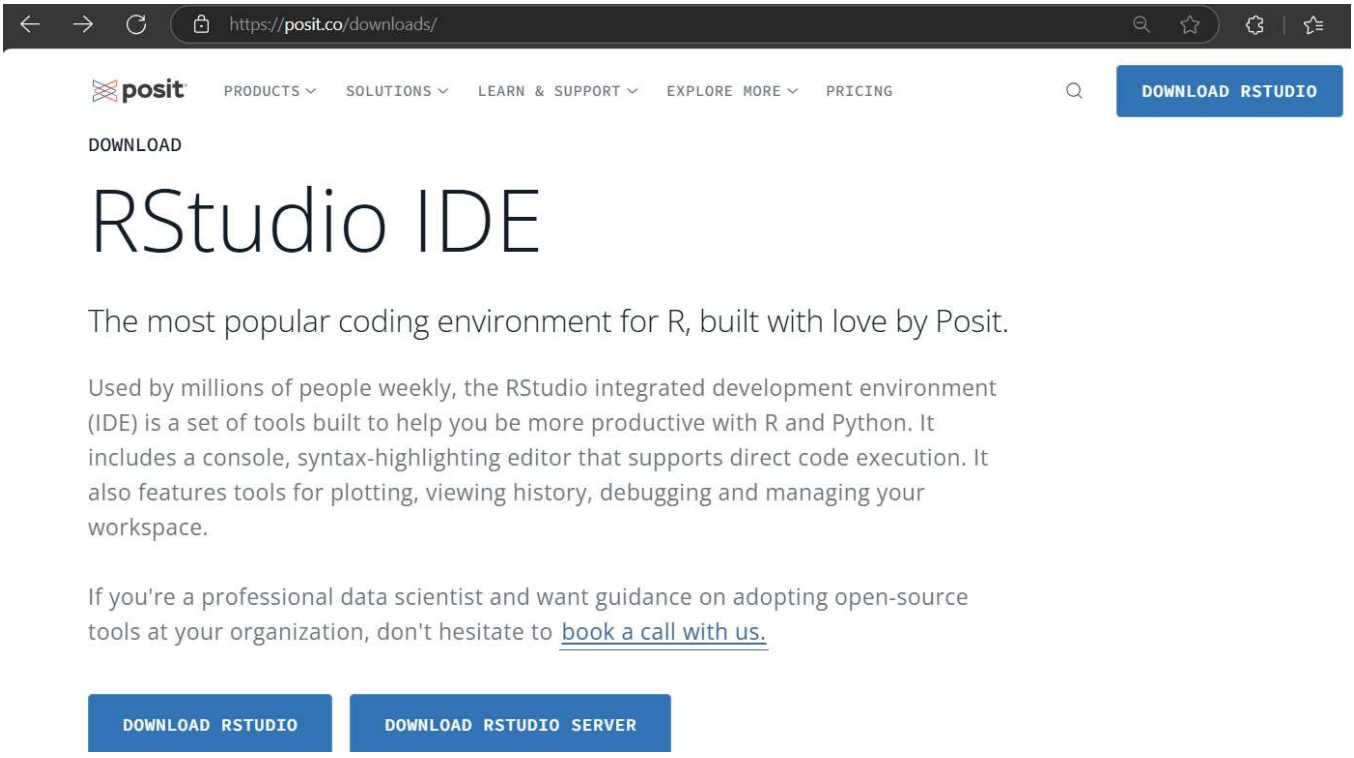
Follow https://cran.r-project.org/ to install it.



2.2 R Studio

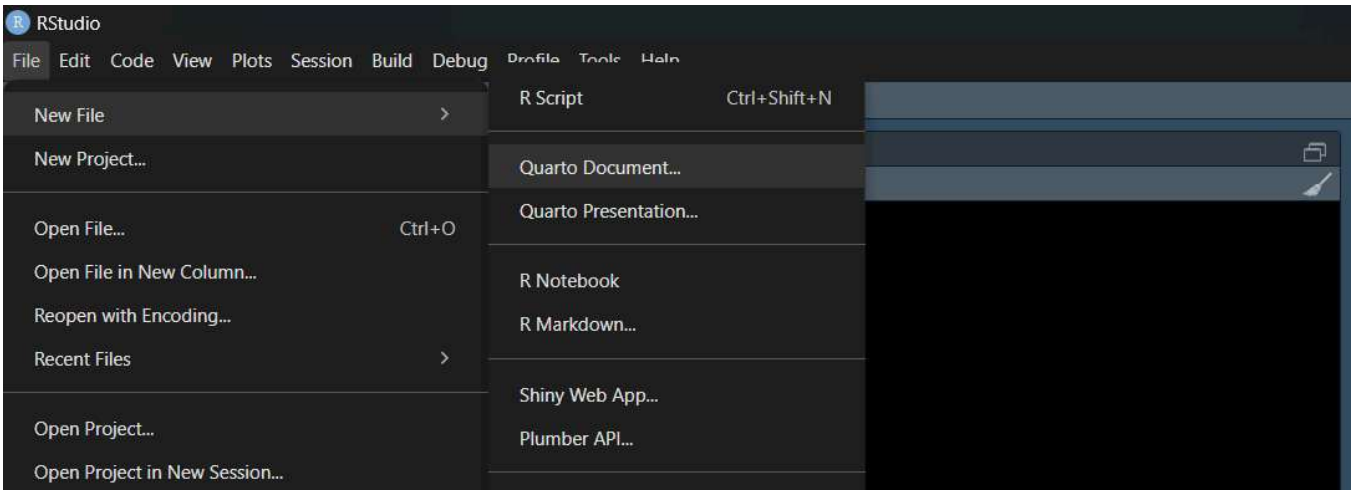
> "RStudio is an integrated development environment (IDE) designed to support multiple languages, including both R and Python." — RStudio

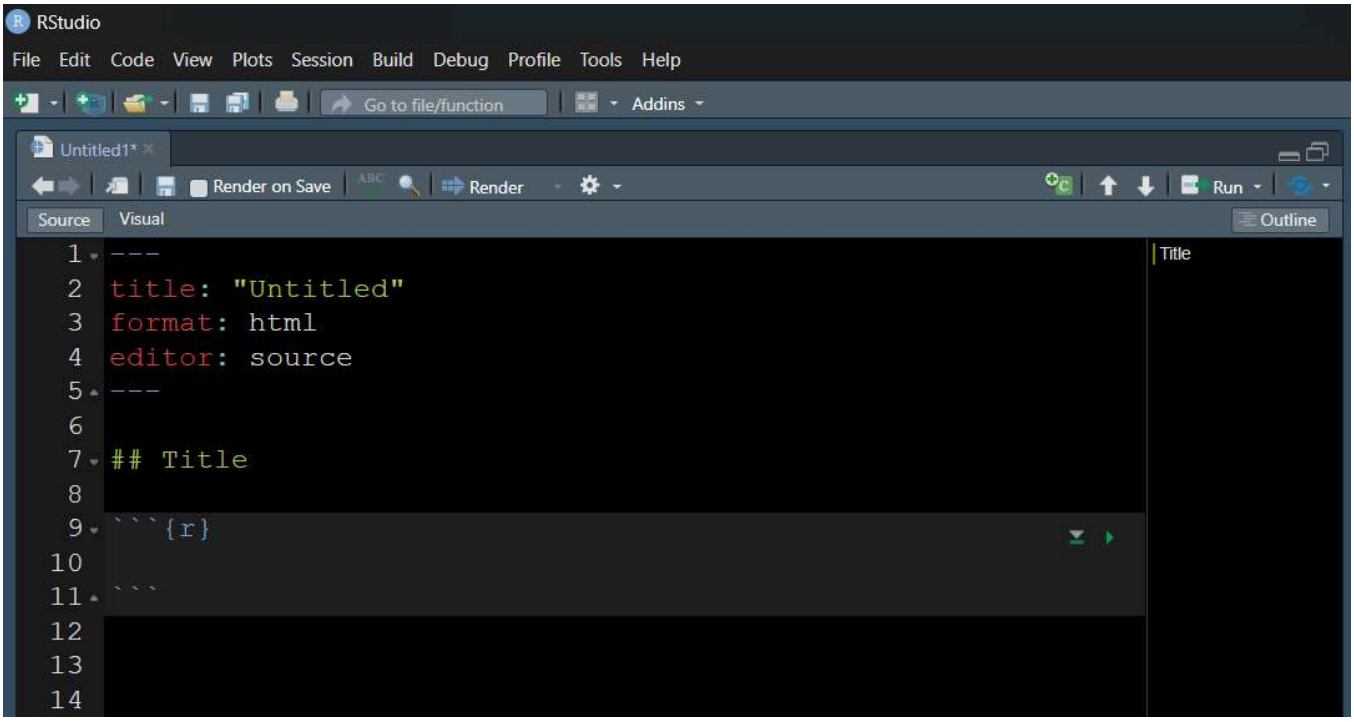
Follow https://posit.co/downloads/ to install it.



2.3 Packages

During the workshop, several R packages will be necessary. Open R Studio and create a new Quarto document.





Insert the following code inside the code chunk.

▼ Code

```
# install.packages("devtools")
library(devtools)
# install.packages("ggcorrplot")
library(ggcorrplot)
# install.packages("glmnet")
library(glmnet)
# install.packages("GCEstim")
library(GCEstim)
```

https://cran.r-project.org/web/packages/GCEstim/index.html

GCEstim: Regression Coefficients Estimation Using the Generalized Cross Entropy

Estimation and inference using the Generalized Maximum Entropy (GME) and Generalized Cross Entropy (GCE) framework, a flexible method for solving ill-posed inverse problems and parameter estimation under uncertainty (Golan, Judge, and Miller (1996, ISBN:978-0471145925) "Maximum Entropy Econometrics: Robust Estimation with Limited Data"). The package includes routines for generalized cross entropy estimation of linear models including the implementation of a GME-GCE two steps approach. Diagnostic tools, and options to incorporate prior information through support and prior distributions are available (Macedo, Cabral, Afreixo, Macedo and Angelelli (2025) <doi:10.1007/978-3-031-97589-9_21>). In particular, support spaces can be defined by the user or be internally computed based on the ridge trace or on the distribution of standardized regression coefficients. Different optimization methods for the objective function can be used. An adaptation of the normalized entropy aggregation (Macedo and Costa (2019) <doi:10.1007/978-3-030-26036-1_2> "Normalized entropy aggregation for inhomogeneous large-scale data") and a two-stage maximum entropy approach for time series regression (Macedo (2022) <doi:10.1080/03610918.2022.2057540>) are also available. Suitable for applications in econometrics, health, signal processing, and other fields requiring robust estimation under data constraints.

Version:	0.1.0
Depends:	R (≥ 3.5.0), zoo
Imports:	downlit, data.table, rlang, lbfgs, lbfgsb3c, meboot, optimParallel, optimx, rstudioapi, stats, clusterGeneration, simstudy, pracma, pathviewr, Rsolnp, bayestestR, ggplot2, ggpubr, ggdist, latex2exp, plotly, viridis, hdrcde, shiny, miniUI, shinyWidgets, shinydashboardPlus, readxl, DT, magrittr
Suggests:	knitr, rmarkdown, kableExtra
Published:	2025-07-16
DOI:	10.32614/CRAN.package.GCEstim
Author:	Cabral Jorge [aut, cre], Macedo Pedro [ths], Afreixo Vera [ths]
Maintainer:	Cabral Jorge <jorgecabral at ua.pt>
BugReports:	https://github.com/jorgevazcabral/GCEstim/issues
License:	GPL-3
URL:	https://github.com/jorgevazcabral/GCEstim
NeedsCompilation:	no
Materials:	README, NEWS
CRAN checks:	GCEstim results

Documentation:

Reference manual:	GCEstim.html , GCEstim.pdf
Vignettes:	Quick start (source, R code) Generalized Maximum Entropy framework (source, R code) Generalized Cross Entropy framework (source, R code) Optimization methods (source, R code) Choosing the supports spaces (source, R code) Two steps ME estimation (source, R code) Further considerations (source, R code)

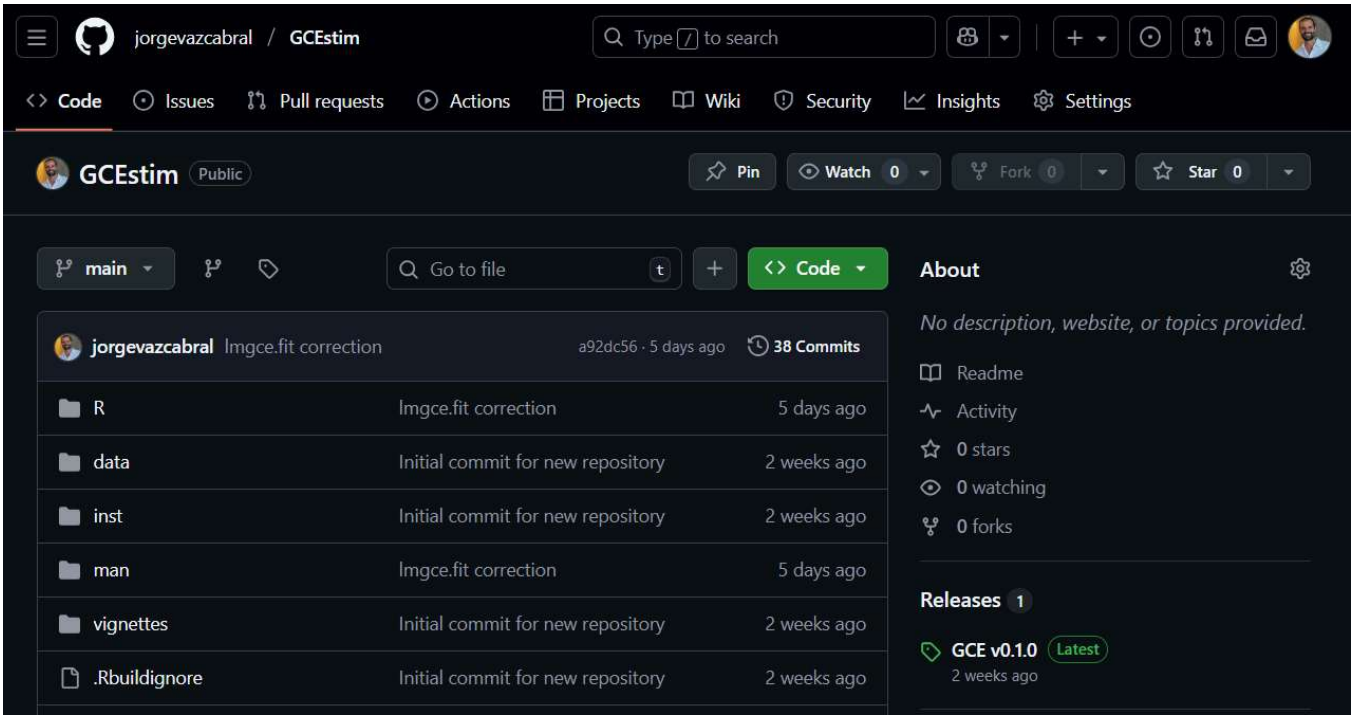
Downloads:

Package source:	GCEstim_0.1.0.tar.gz
Windows binaries:	r-devel: GCEstim_0.1.0.zip, r-release: GCEstim_0.1.0.zip, r-oldrel: GCEstim_0.1.0.zip
macOS binaries:	r-release (arm64): GCEstim_0.1.0.tgz, r-oldrel (arm64): GCEstim_0.1.0.tgz, r-release (x86_64): GCEstim_0.1.0.tgz, r-oldrel (x86_64): GCEstim_0.1.0.tgz

Linking:

Please use the canonical form https://CRAN.R-project.org/package=GCEstim to link to this page.

https://github.com/jorgevazcabral



3 Simulated Data

3.1 Low collinearity

3.1.1 Generate Data

▼ Code

```
n = 100 # Number of individuals
intercept.beta = 0 # Intercept
y.gen.cont.beta = c(3, 6, 9) # Coefficients used for generating y
cont.k = 0 # Number of noisy continuous variables
condnumber = 1 # Condition number of the X matrix
seed <- 1357

simulated_data_coef <-
  c(intercept.beta,
    rep(0, cont.k),
    y.gen.cont.beta)
```

▼ Code

```
simulated_data <-
  fngendata(
    n = n,
    cont.k = cont.k,
    y.gen.cont.beta = y.gen.cont.beta,
    intercept.beta = intercept.beta,
```

```
    Xgenerator.method = "svd",
    condnumber = condnumber,
    seed = seed)

prop_train = 0.7

set.seed(seed)
ind <- sample(1:n,
              floor(prop_train * n))

simulated_data_train <- simulated_data[ind,]
simulated_data_test <- simulated_data[-ind,]
```

3.1.1.1 Summary

▼ Code

```
summary(simulated_data_train)
```

```
      X001                X002                 X003                  y          
 Min.   :-0.29293   Min.   :-0.170424   Min.   :-0.238780   Min.   :-3.6779  
 1st Qu.:-0.07523   1st Qu.:-0.062134   1st Qu.:-0.054976   1st Qu.:-1.0064  
 Median :-0.02681   Median :-0.006911   Median :-0.005206   Median :-0.1649  
 Mean   :-0.01555   Mean   :-0.007363   Mean   : 0.011149   Mean   :-0.2056  
 3rd Qu.: 0.05892   3rd Qu.: 0.053372   3rd Qu.: 0.078745   3rd Qu.: 0.5678  
 Max.   : 0.19954   Max.   : 0.209718   Max.   : 0.209840   Max.   : 2.9427  
```

3.1.1.2 Correlation matrix

ALL | TRAIN | TEST

▼ Code

```
ggcorrplot::ggcorrplot(
  cor(simulated_data),
  lab = TRUE,
  title = paste("Cond =",
                round(base::kappa(
                  simulated_data[,-ncol(simulated_data)],
                  exact = TRUE),
                  0)))
```

3.1.2 Estimation

3.1.2.1 OLS

▼ Code

```
sd_model_OLS <-
  lm(data = simulated_data_train,
     y ~ . # or y ~ X001 + X002 + X003
     )
```

▼ Code

```
# summary of the linear model
summary(sd_model_OLS)
```

```
Call:
lm(formula = y ~ ., data = simulated_data_train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.60148 -0.62532  0.02242  0.55744  1.58774

Coefficients:
            Estimate Std. Error t value         Pr(>|t|)
```

```
(Intercept)  -0.2398     0.1005  -2.385           0.0199 *
X001          1.3376     0.9521   1.405           0.1647
X002          5.5875     1.1029   5.066 0.00000349859864 ***
X003          8.6183     1.0169   8.475 0.00000000000374 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8221 on 66 degrees of freedom
Multiple R-squared:  0.6092,    Adjusted R-squared:  0.5914
F-statistic: 34.29 on 3 and 66 DF,  p-value: 0.0000000000001772
```

▼ Code

```
# coefficients of the linear model
sd_model_OLS_coef <- coef(sd_model_OLS)
```

▼ Code

```
# prediction error RMSE
(sd_model_OLS_RMSE_train <-
   accmeasure(fitted(sd_model_OLS),
              simulated_data_train$y))
```

```
[1] 0.7982673
```

▼ Code

```
(sd_model_OLS_RMSE_test <-
   accmeasure(predict(sd_model_OLS,
                simulated_data_test),
              simulated_data_test$y))
```

```
[1] 1.083268
```

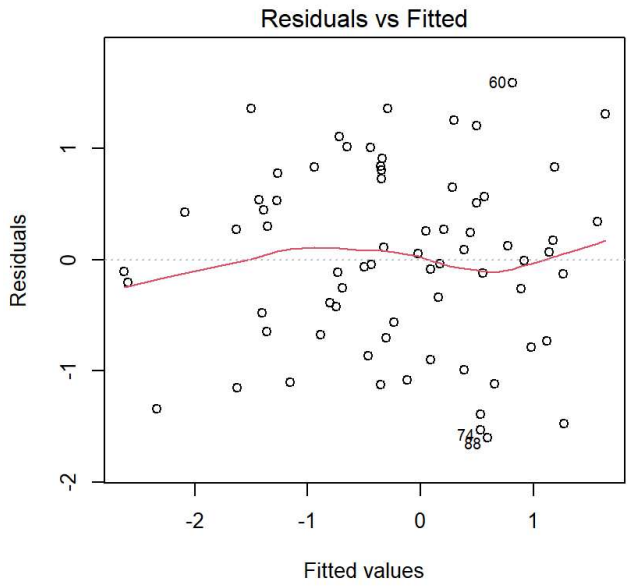
▼ Code

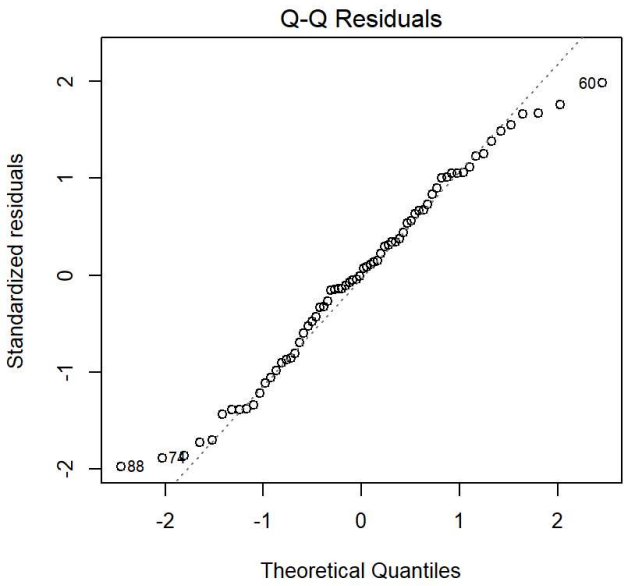
```
# precision error RMSE
(sd_model_OLS_beta_RMSE <-
    accmeasure(sd_model_OLS_coef,
               simulated_data_coef))
```

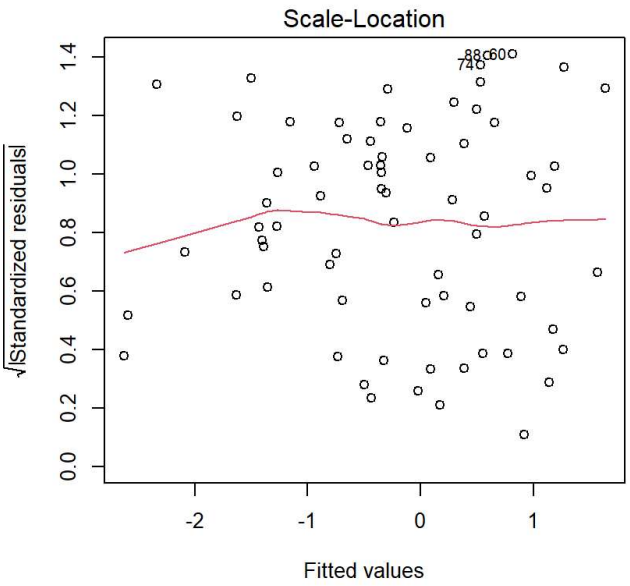
```
[1] 0.8855462
```

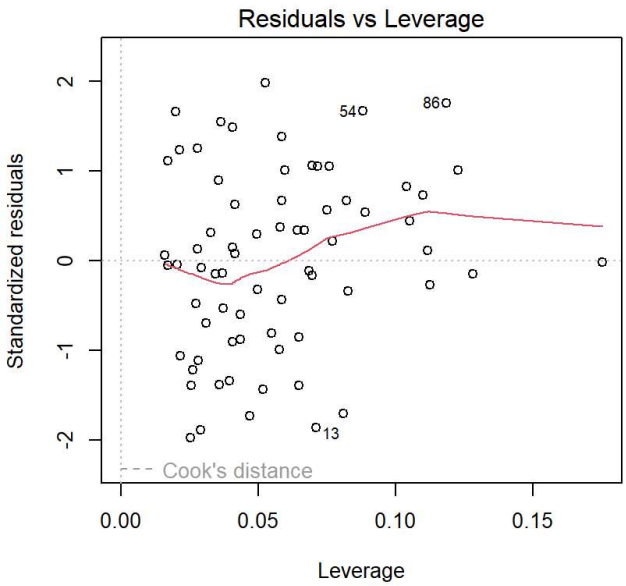
▼ Code

```
par(mfrow = c(2,2))
plot(sd_model_OLS)
```









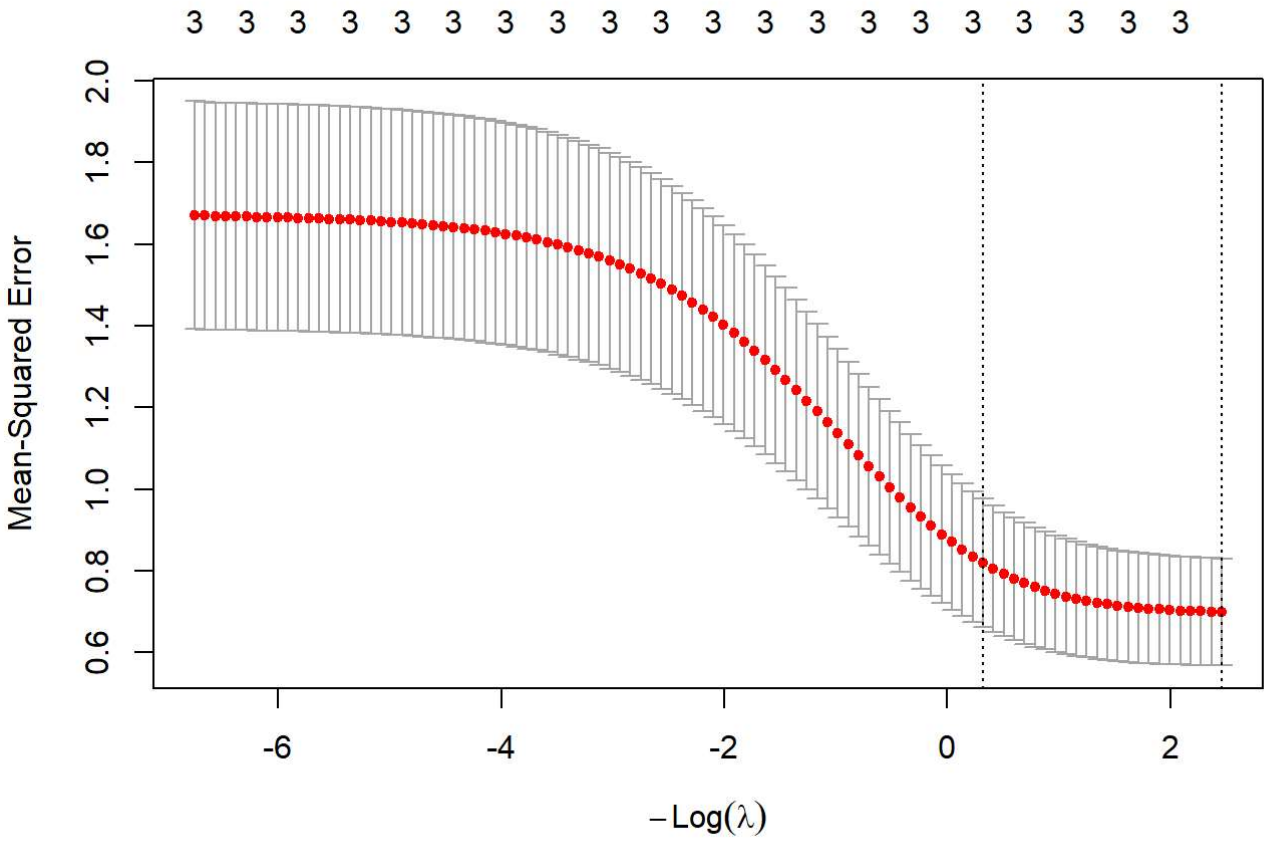
3.1.2.2 Ridge

▼ Code

```
set.seed(seed)
sd_model_ridge <-
  glmnet::cv.glmnet(
    x = as.matrix(simulated_data_train[,-ncol(simulated_data_train)]),
    y = simulated_data_train[,ncol(simulated_data_train)],
    alpha = 0)
```

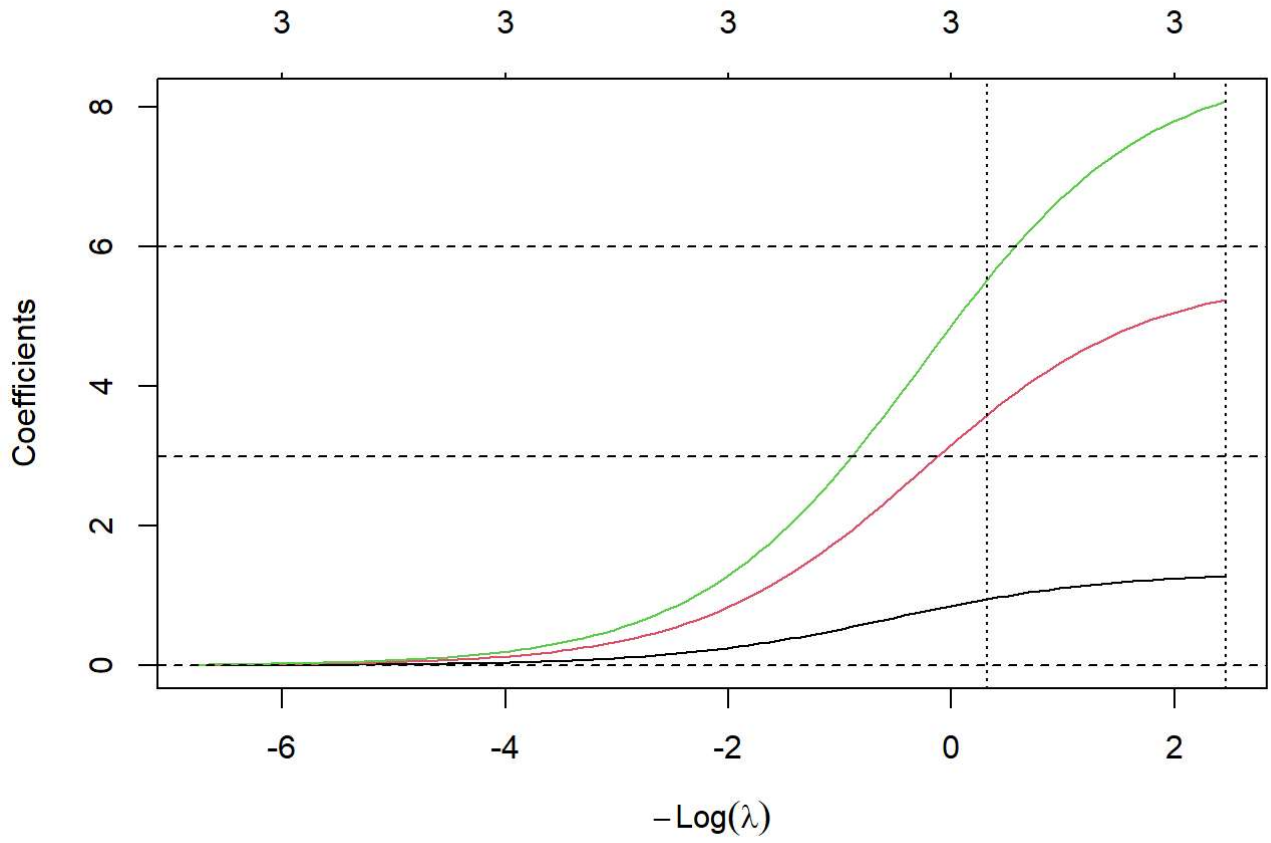
▼ Code

```
plot(sd_model_ridge)
```



▼ Code

```
plot(sd_model_ridge$glmnet.fit,
     xvar = "lambda")
abline(v = -c(log(sd_model_ridge$lambda.min),
              log(sd_model_ridge$lambda.1se)),
       lty = 3)
abline(h = simulated_data_coef,
       lty = 2)
```



▼ Code

```
(sd_model_ridge_coef <-
  t(as.matrix(coef(sd_model_ridge,
                   s = "lambda.min")))) # s = "lambda.1se"
```

```
           (Intercept)     X001     X002     X003
lambda.min  -0.2372867 1.279058 5.245306 8.087534
```

▼ Code

```
(sd_model_ridge_RMSE_train <-
  accmeasure(predict(sd_model_ridge,
                     as.matrix(simulated_data_train[,-ncol(simulated_data_train)]),
                     s = "lambda.min"),
       simulated_data_train$y))
```

```
[1] 0.8005858
```

▼ Code

```
(sd_model_ridge_RMSE_test <-
  accmeasure(predict(sd_model_ridge,
                     as.matrix(simulated_data_test[,-ncol(simulated_data_test)]),
```

```
                    s = "lambda.min"),
              simulated_data_test$y))
```

```
[1] 1.076906
```

▼ Code

```
 (sd_model_ridge_beta_RMSE <-
   accmeasure(sd_model_ridge_coef,
              simulated_data_coef))
```

```
[1] 1.051202
```

3.1.2.3 GCE

▼ Code

```
 #help(lmgce)
```

▼ Code

```
 sd_model_GCE <-
   lmgce(data = simulated_data_train,
         y ~ .,
         boot.B = 100,
         seed = seed
       )
```

▼ Code

```
 summary(sd_model_GCE)
```

```
Call:
lmgce(formula = y ~ ., data = simulated_data_train, boot.B = 100,
    seed = seed)

Residuals:
    Min      1Q  Median      3Q     Max
-1.6763 -0.7201 -0.0201  0.5189  1.5030

Coefficients:
            Estimate Std. Deviation z value             Pr(>|t|)
(Intercept) -0.15525        0.04845  -3.204              0.00135 **
X001         1.47590        0.45889   3.216              0.00130 **
X002         5.67417        0.53156  10.675 < 0.0000000000000002 ***
X003         8.45006        0.49010  17.242 < 0.0000000000000002 ***
---
```

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

Normalized Entropy:
             NormEnt  SupportLL SupportUL
(Intercept) 0.930529  -0.470056  0.470056
X001        0.997119 -21.687070 21.687070
X002        0.968101 -25.183138 25.183138
X003        0.914806 -23.168654 23.168654

Residual standard error: 0.8265 on 66 degrees of freedom
Chosen Upper Limit for Standardized Supports: 1.7583, Chosen Error: 1se
Multiple R-squared: 0.6032, Adjusted R-squared: 0.5851
NormEnt: 0.9526, CV-NormEnt: 0.9544 (0.002983)
RMSE: 0.8026, CV-RMSE: 0.8323 (0.06963)
```

▼ Code

```
sd_model_GCE_coef <- coef(sd_model_GCE)
```

▼ Code

```
(sd_model_GCE_RMSE_train <-
  accmeasure(fitted(sd_model_GCE),
             simulated_data_train$y))
```

```
[1] 0.802554
```

▼ Code

```
(sd_model_GCE_RMSE_test <-
    accmeasure(predict(sd_model_GCE,
                       simulated_data_test),
               simulated_data_test$y))
```

```
[1] 1.063212
```

▼ Code

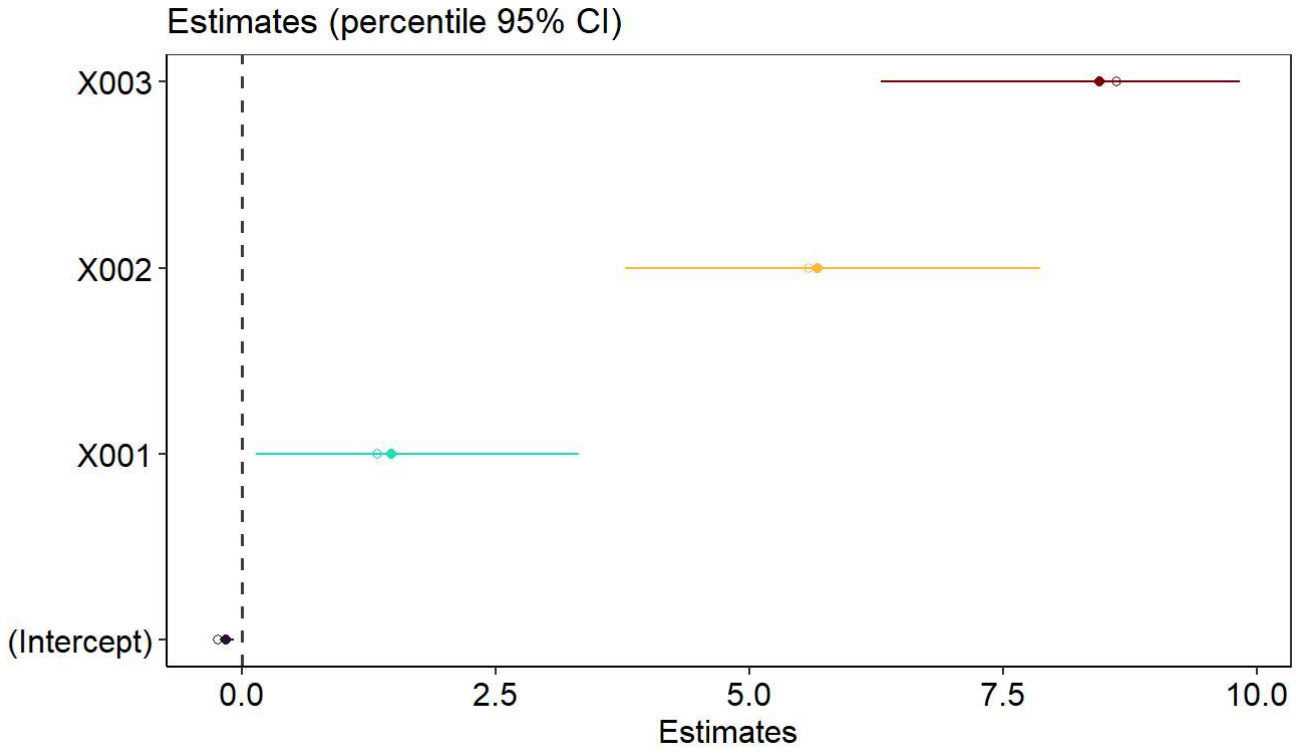
```
(sd_model_GCE_beta_RMSE <-
  accmeasure(sd_model_GCE_coef,
             simulated_data_coef))
```

```
[1] 0.8299966
```

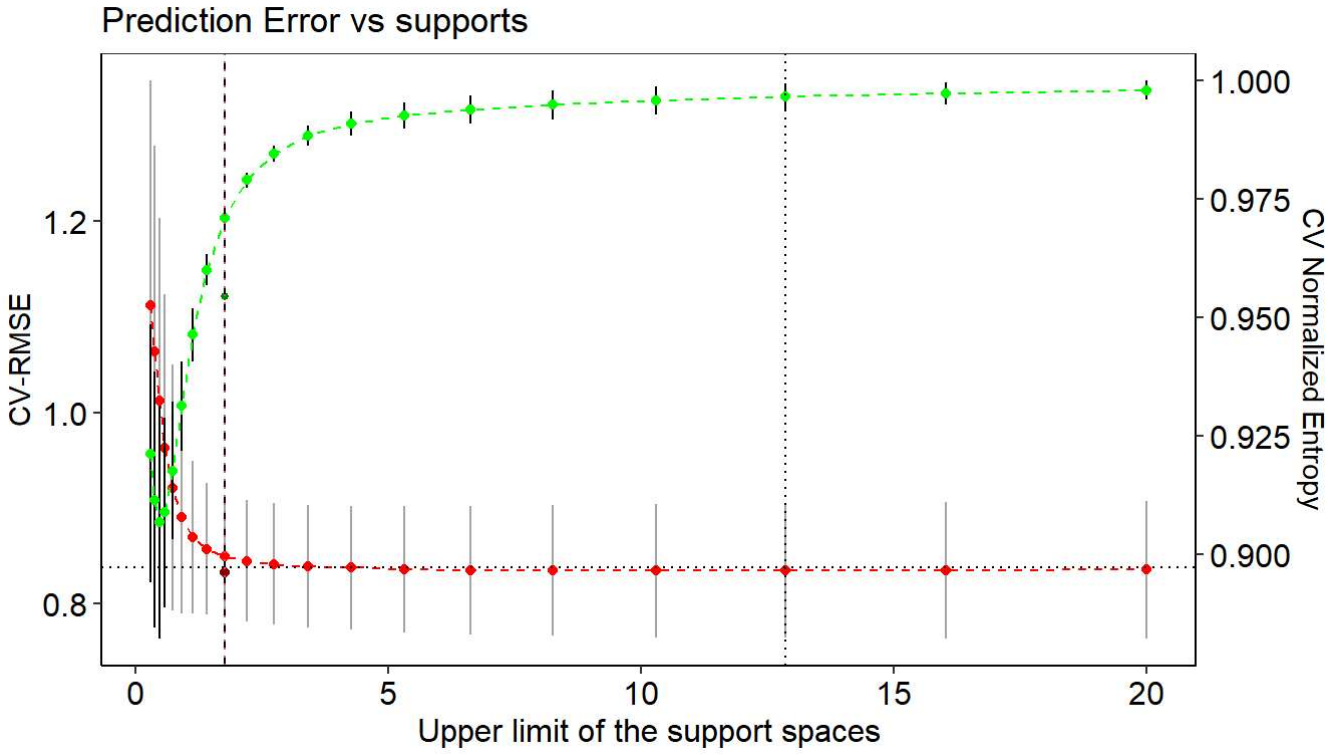
▼ Code

```
plot(sd_model_GCE, ci.method = "percentile")
```

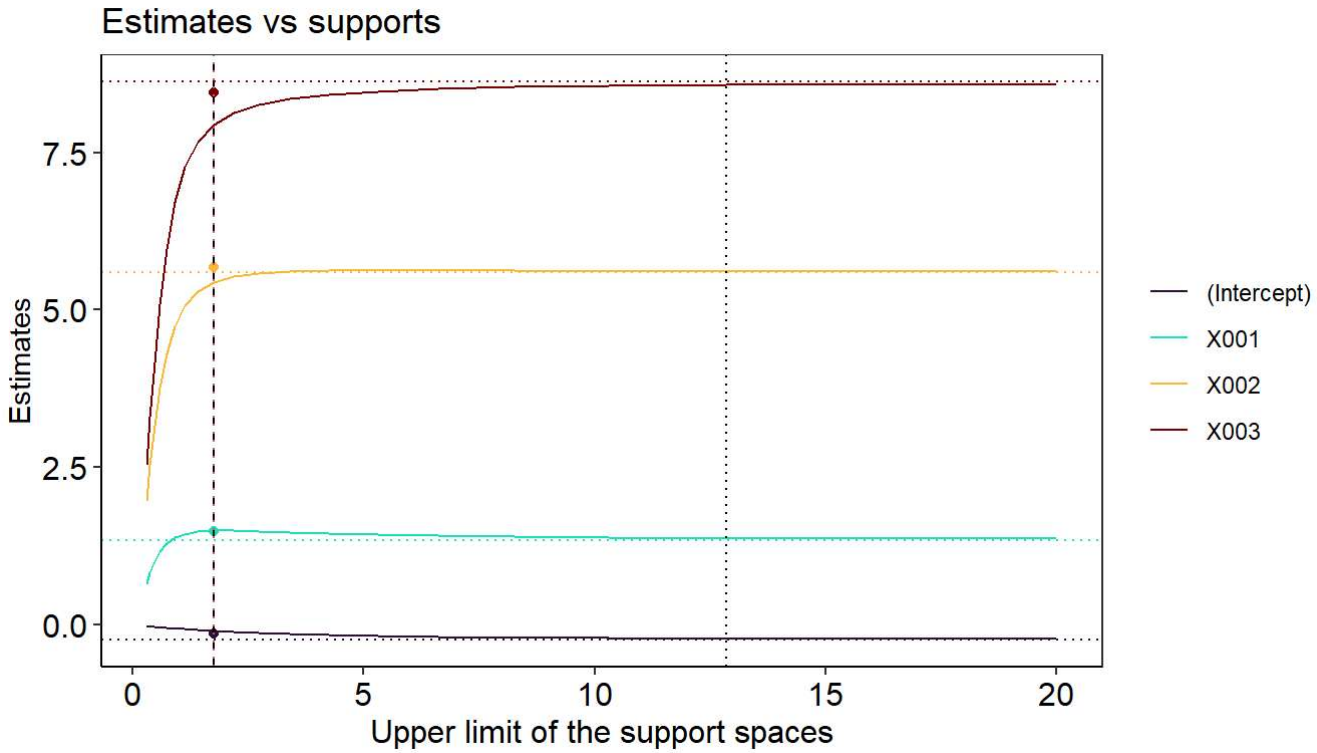
```
$p1
```


Estimates (percentile 95% CI)
X003
X002
X001
(Intercept)
0.0
2.5
5.0
7.5
10.0
Estimates

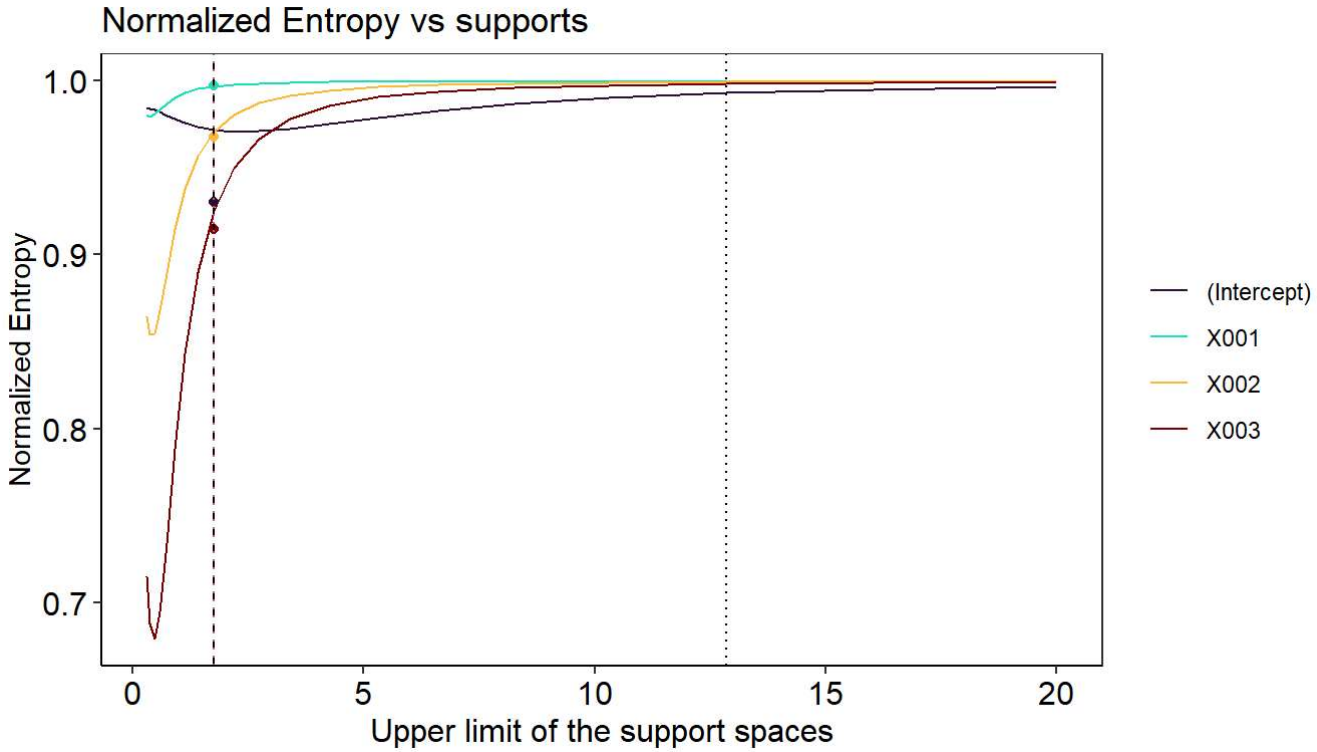
$p2


Prediction Error vs supports
CV-RMSE
1.2
1.0
0.8
1.000
0.975
0.950
0.925
0.900
CV Normalized Entropy
0
5
10
15
20
Upper limit of the support spaces

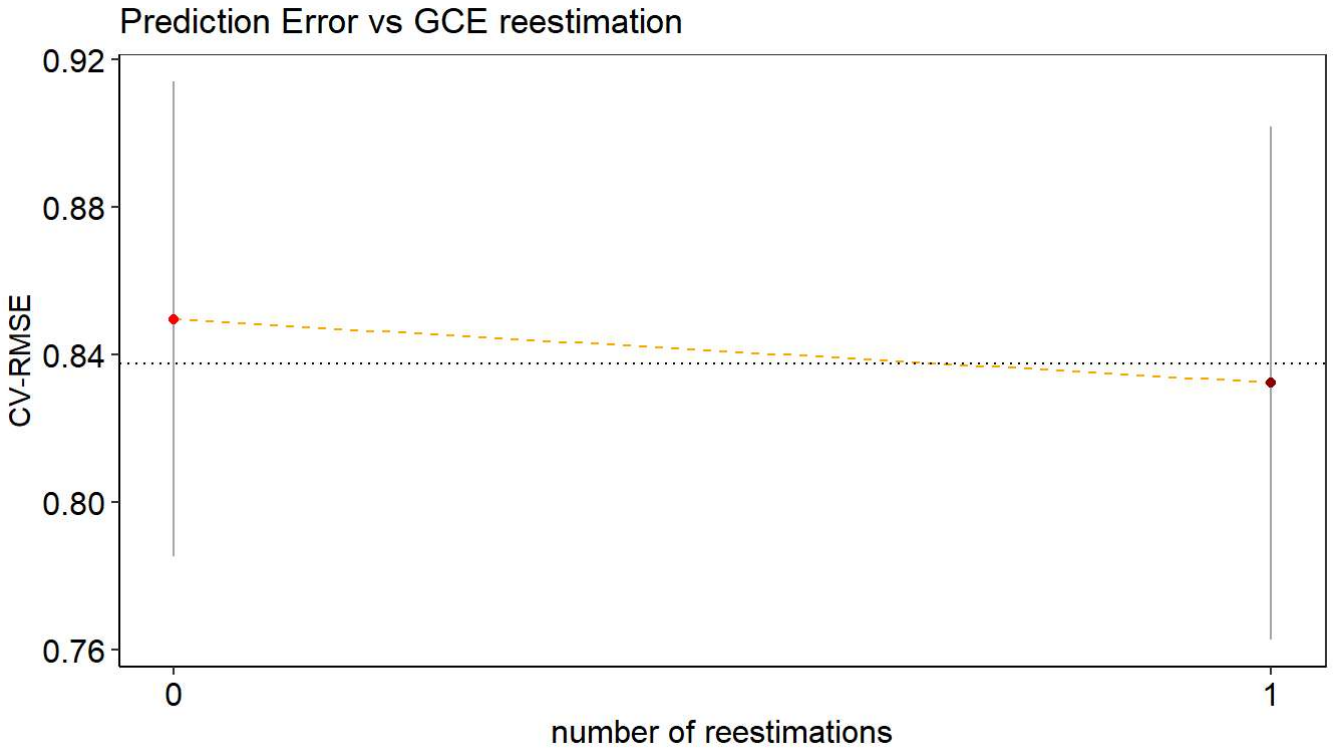
$p3


Estimates vs supports
Estimates
7.5
5.0
2.5
0.0
0
5
10
15
20
Upper limit of the support spaces
(Intercept)
X001
X002
X003

$p4


Normalized Entropy vs supports
Normalized Entropy
1.0
0.9
0.8
0.7
0
5
10
15
20
Upper limit of the support spaces
(Intercept)
X001
X002
X003

$p6



3.1.2.4 All

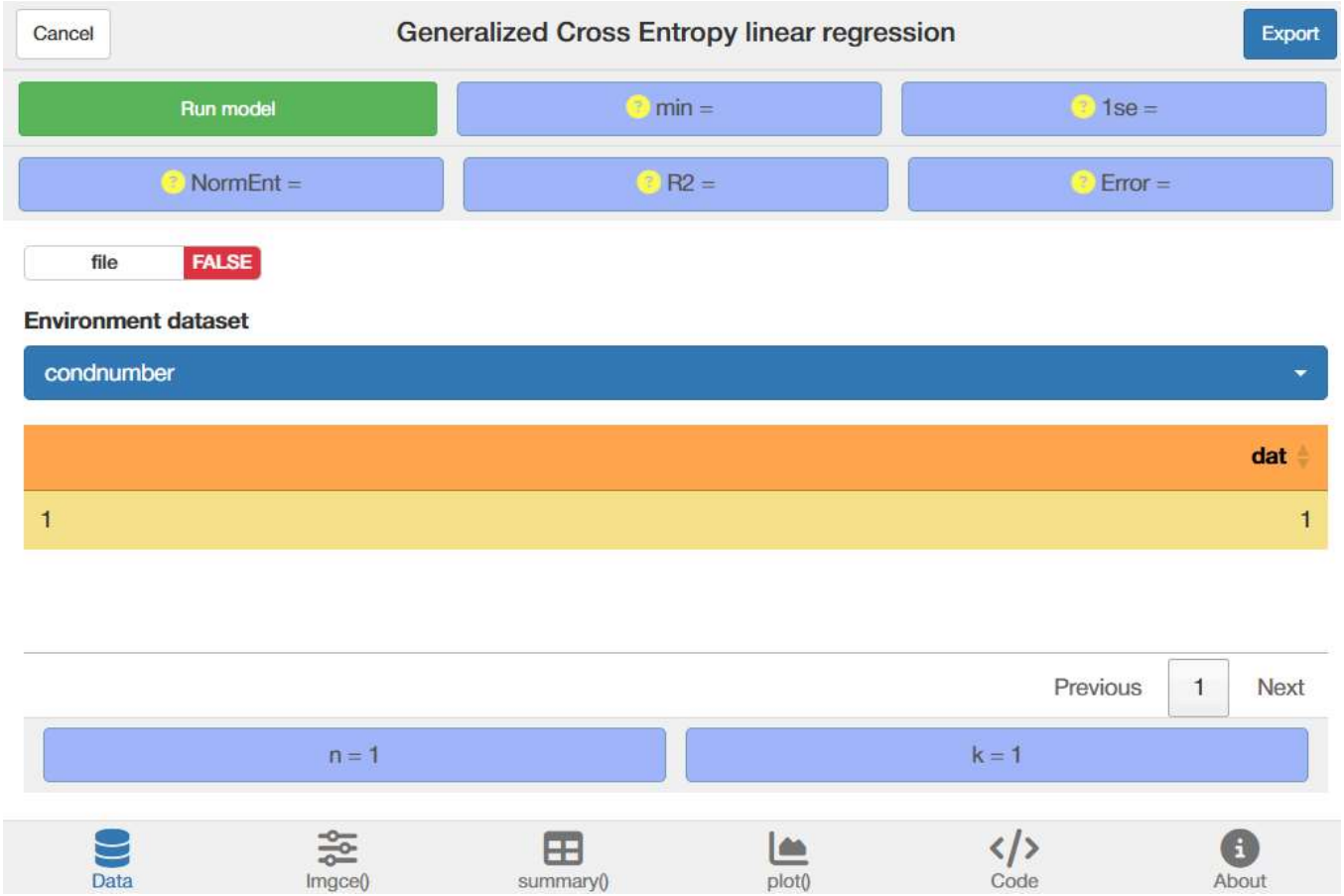
Show 25 entries Search:

	Generating	OLS	ridge
(Intercept)	0	-0.23977	-0.23729	-0.15525
X001	3	1.33765	1.27906	1.4759
X002	6	5.58746	5.24531	5.67417
X003	9	8.61834	8.08753	8.45006
beta_RMSE	0	0.88555	1.0512	0.83
RMSE_train	0	0.79827	0.80059	0.80255
RMSE_test	0	1.08327	1.07691	1.06321

Showing 1 to 7 of 7 entries Previous 1 Next

4 Web R Shiny App

▶ Code



5 Acknowledgments

This workshop is supported by the Center for Research and Development in Mathematics and Applications (CIDMA) under the Portuguese Foundation for Science and Technology (FCT, https://ror.org/00snfqn58) Multi-Annual Financing Program for R&D Units, grants UID/4106/2025 and UID/PRR/4106/2025.

References

Bring, Johan. 1994. “How to Standardize Regression Coefficients.” *The American Statistician* 48: 209–13. https://doi.org/10.2307/2684719.

Caputo, Michael R, and Quirino Paris. 2008. “Comparative Statics of the Generalized Maximum Entropy Estimator of the General Linear Model.” *European Journal of Operational Research* 185: 195–203. https://doi.org/https://doi.org/10.1016/j.ejor.2006.12.031.

Golan, Amos, George G. Judge, and Douglas Miller. 1996. *Maximum Entropy Econometrics : Robust Estimation with Limited Data*. Wiley.

Greene, W H. 2008. *Econometric Analysis*. Pearson/Prentice Hall. https://books.google.pt/books?id=b541vgAACAAJ.

Hoerl, Arthur E, and Robert W Kennard. 1970. “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” *Technometrics* 12 (February): 55–67. https://doi.org/10.1080/00401706.1970.10488634.

Jaynes, E T. 1957. “Information Theory and Statistical Mechanics.” *Physical Review* 106 (May): 620–30. https://doi.org/10.1103/PhysRev.106.620.

Pukelsheim, Friedrich. 1994. “The Three Sigma Rule.” *The American Statistician* 48: 88–91. https://doi.org/10.2307/2684253.

RCoreTeam. 2025. “R: A Language and Environment for Statistical Computing.” https://www.R-project.org/.