

Package ‘estimability’

June 26, 2026

Type Package

Title Tools for Assessing Estimability of Linear Predictions

Version 2.0.0

Date 2026-06-18

Depends stats, R(>= 4.1.0)

Suggests knitr, rmarkdown

Description Provides tools for determining estimability of linear functions of regression coefficients, and 'epredict' methods that handle non-estimable cases correctly. Estimability theory is discussed in many linear-models textbooks including Chapter 3 of Monahan, JF (2008), ``A Primer on Linear Models'', Chapman and Hall (ISBN 978-1-4200-6201-4).

URL <https://github.com/rvlenth/estimability>,
<https://rvlenth.github.io/estimability/>

BugReports <https://github.com/rvlenth/estimability/issues>

ByteCompile yes

License GPL (>= 3)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Russell Lenth [aut, cph],
Michael Dumelle [cre, aut]

Maintainer Michael Dumelle <Dumelle.Michael@epa.gov>

Repository CRAN

Date/Publication 2026-06-26 08:10:02 UTC

Contents

estimability-package	2
epredict	3
estble.subspace	6
nonest.basis	7
Index	11

estimability-package	<i>Estimability Tools for Linear Models</i>
----------------------	---

Description

Provides tools for determining estimability of linear functions of regression coefficients, and alternative epredict methods for lm, glm, and mlm objects that handle non-estimable cases correctly.

Details

Package: estimability
Type: Package
Details: See DESCRIPTION file

When a linear model is not of full rank, the regression coefficients are not uniquely estimable. However, the predicted values are unique, as are other linear combinations where the coefficients lie in the row space of the data matrix. Thus, estimability of a linear function of regression coefficients can be determined by testing whether the coefficients lie in this row space – or equivalently, are orthogonal to the corresponding null space.

This package provides functions `nonest.basis` and `is.estble` to facilitate such an estimability test. Package developers may find these useful for incorporating in their predict methods when new predictor settings are involved.

The function `estble.subspace` is useful for projecting a matrix onto an estimable subspace whose rows are all estimable.

The package also provides `epredict` methods – alternatives to the `predict` methods in the `stats` package for "lm", "glm", and "mlm" objects. When the `newdata` argument is specified, estimability of each new prediction is checked and any non-estimable cases are replaced by NA.

Author(s)

Russell V. Lenth <russell-lenth@uiowa.edu>

References

Monahan, John F. (2008) *A Primer on Linear Models*, CRC Press. (Chapter 3)

Description

Estimability Enhancements for lm and Relatives

Usage

```
epredict(object, ...)  
  
## S3 method for class 'lm'  
epredict(  
  object,  
  newdata,  
  ...,  
  type = c("response", "terms", "matrix", "estimability"),  
  nonest.tol = 1e-08,  
  nbasis = object$nonest  
)  
  
## S3 method for class 'glm'  
epredict(  
  object,  
  newdata,  
  ...,  
  type = c("link", "response", "terms", "matrix", "estimability"),  
  nonest.tol = 1e-08,  
  nbasis = object$nonest  
)  
  
## S3 method for class 'mlm'  
epredict(  
  object,  
  newdata,  
  ...,  
  type = c("response", "matrix", "estimability"),  
  nonest.tol = 1e-08,  
  nbasis = object$nonest  
)  
  
eupdate(object, ...)  
  
## S3 method for class 'lm'  
eupdate(object, ...)
```

Arguments

<code>object</code>	An object inheriting from <code>lm</code>
<code>...</code>	Arguments passed to <code>predict</code> or <code>update</code>
<code>newdata</code>	A <code>data.frame</code> containing predictor combinations for new predictions
<code>type</code>	character string specifying the desired result. See Details.
<code>nonest.tol</code>	Tolerance used by <code>is.estble</code> to check estimability of new predictions
<code>nbasis</code>	a basis for the null space, e.g., a result of a call to <code>nonest.basis</code> . If <code>nbasis</code> is <code>NULL</code> , a basis is constructed from <code>object</code> .

Details

These functions call the corresponding S3 `predict` methods in the **stats** package, but with a check for estimability of new predictions, and with appropriate actions for non-estimable cases.

If `newdata` is missing or `object` is not rank-deficient, this method directly to the same method in the **stats** library. In rank-deficient cases with `newdata` provided, each row of `newdata` is tested for estimability against the null basis provided in `nbasis`. Any non-estimable cases found are replaced with NAs.

The `type` argument is passed to `predict` when it is one of "response", "link", or "terms". With `newdata` present and `type = "matrix"`, the model matrix for `newdata` is returned, with an attribute "estble" that is a logical vector of length `nrow(newdata)` indicating whether each row is estimable. With `type = "estimability"`, just the logical vector is returned.

If you anticipate making several `epredict` calls with new data, it improves efficiency to either obtain the null basis and provide it in the call, or add it to `object` with the name "nonest" (perhaps via a call to `update`).

`update` is an S3 generic function with a method provided for "lm". It updates the object according to any arguments in `...`, then obtains the updated object's nonestimable basis and returns it in `object$nonest`.

Value

The same as the result of a call to the `predict` method in the **stats** package, except rows or elements corresponding to non-estimable predictor combinations are set to NA. The value for `type` is "matrix" or "estimability" is explained under details.

Note

The capabilities of the `epredict` function for `lm` objects is provided by `predict.lm` (if using R version 4.3.0 or later) with `rankdeficient = "NA"`; however, `epredict` uses **estimability**'s own criteria to determine which predictions are set to NA. An advantage of using `epredict` is one of efficiency: we can compute the null basis once and for all and have it available for additional predictions, whereas `predict.lm` will re-compute it each time. If the user wishes to see a message explaining why NAs were displayed, set `options(estimability.verbose = TRUE)`.

Examples

```

require("estimability")

# Fake data where x3 and x4 depend on x1, x2, and intercept
x1 <- -4:4
x2 <- c(-2,1,-1,2,0,2,-1,1,-2)
x3 <- 3*x1 - 2*x2
x4 <- x2 - x1 + 4
y <- 1 + x1 + x2 + x3 + x4 + c(-.5,.5,.5,-.5,0,.5,-.5,-.5,.5)

# Different orderings of predictors produce different solutions
mod1234 <- lm(y ~ x1 + x2 + x3 + x4)
mod4321 <- eupdate(lm(y ~ x4 + x3 + x2 + x1))
# (Estimability checking with mod4321 will be more efficient because
# it will not need to recreate the basis)
mod4321$nonest

# test data:
testset <- data.frame(
  x1 = c(3, 6, 6, 0, 0, 1),
  x2 = c(1, 2, 2, 0, 0, 2),
  x3 = c(7, 14, 14, 0, 0, 3),
  x4 = c(2, 4, 0, 4, 0, 4))

# Look at predictions when we don't check estimability
suppressWarnings( # Disable the warning from stats::predict.lm
  rbind(p1234 = predict(mod1234, newdata = testset),
        p4321 = predict(mod4321, newdata = testset)))

# Compare with results when we do check:
rbind(p1234 = epredict(mod1234, newdata = testset),
      p4321 = epredict(mod4321, newdata = testset))

# stats::predict has same capability for lm objects starting in version 4.3.0:
if((R.Version())$major >= 4) && (R.Version())$minor >= 3))
  stats::predict(mod1234, newdata = testset, rankdeficient = "NA")

# Note that estimable cases have the same predictions

# change mod1234 and include nonest basis
mod134 <- eupdate(mod1234, . ~ . - x2, subset = -c(3, 7))
mod134$nonest

# When row spaces are the same, bases are interchangeable
# so long as you account for the ordering of parameters:
epredict(mod4321, newdata = testset, type = "estimability",
  nbasis = nonest.basis(mod1234)[c(1,5:2), ])

# Comparison with predict.lm in R >= 4.3.0
if((R.Version())$major >= 4) && (R.Version())$minor >= 3))
  stats::predict(mod4321, newdata = testset, rankdeficient = "NA")

```

```
## Not run:
### Additional illustration
example(nonest.basis) ## creates model objects warp.lm1 and warp.lm2

# The two models have different contrast specs. But the empty cell
# is correctly identified in both:
fac.cmb <- expand.grid(wool = c("A", "B"), tension = c("L", "M", "H"))
cbind(fac.cmb,
      pred1 = epredict(warp.lm1, newdata = fac.cmb),
      pred2 = epredict(warp.lm2, newdata = fac.cmb))

## End(Not run)
```

estble.subspace	<i>Find an estimable subspace</i>
-----------------	-----------------------------------

Description

Determine a transformation B of the rows of a matrix L such that $B \%*\% L$ is estimable. A practical example is in jointly testing a set of contrasts L in a linear model, and we need to restrict to the subspace spanned by the rows of L that are estimable.

Usage

```
estble.subspace(L, nbasis, tol = 1e-08)
```

Arguments

<code>L</code>	A matrix of dimensions k by p
<code>nbasis</code>	A k by b matrix whose columns form a basis for non-estimable linear functions – such as is returned by nonest.basis
<code>tol</code>	Numeric tolerance for assessing nonestimability. See is.estble .

Details

We require B such that all the rows of $M = B \%*\% L$ are estimable, i.e. orthogonal to the columns of $nbasis$. Thus, we need $B \%*\% L \%*\% nbasis$ to be zero, or equivalently, $t(B)$ must be in the null space of $t(L \%*\% nbasis)$. This can be found using [nonest.basis](#).

Value

An r by p matrix $M = B \%*\% L$ whose rows are all orthogonal to the columns of $nbasis$. The matrix B is attached as `attr(M, "B")`. Note that if any rows of L were non-estimable, then r will be less than k . In fact, if there are no estimable functions in the row space of L , then $r = 0$.

Examples

```
### Find a set of estimable interaction contrasts for a 3 x 4 design
### with two empty cells.
des <- expand.grid(A = factor(1:3), B = factor(1:4))
des <- des[-c(5, 12), ] # cells (2,2) and (3,4) are empty

X <- model.matrix(~ A * B, data = des)
N <- nonest.basis(X)

L <- cbind(matrix(0, nrow = 6, ncol = 6), diag(6))
# i.e., give nonzero weight only to interaction effects

estble.subspace(L, N)

# Tougher demo: create a variation where all rows of L are non-estimable
set.seed(1921.0923) # Franklin Graybill's birthday
LL <- matrix(rnorm(36), ncol = 6) %%% L
estble.subspace(LL, N)
```

nonest.basis

Estimability Tools

Description

This documents the functions needed to test estimability of linear functions of regression coefficients.

Usage

```
nonest.basis(x, ...)

## Default S3 method:
nonest.basis(x, ...)

## S3 method for class 'qr'
nonest.basis(x, ...)

## S3 method for class 'matrix'
nonest.basis(
  x,
  tol = 5e-08,
  rank = attr(x, "rank"),
  pivot = attr(x, "pivot"),
  ...
)

## S3 method for class 'lm'
```

```

nonest.basis(x, ...)

## S3 method for class 'svd'
nonest.basis(x, tol = 5e-08, rank = NULL, pivot = NULL, ...)

legacy.nonest.basis(x, ...)

is.estble(x, nbasis, tol = 1e-08)

all.estble

```

Arguments

<code>x</code>	For <code>nonest.basis</code> , an object of a class in <code>'methods("nonest.basis")'</code> . Or, in <code>is.estble</code> , a numeric vector or matrix for assessing estimability of <code>'sum(x * beta)'</code> , where <code>beta</code> is the vector of regression coefficients.
<code>...</code>	Additional arguments passed to other methods.
<code>tol</code>	Numeric tolerance for assessing rank or nonestimability. For determining rank, singular values less than <code>tol</code> times the largest singular value are regarded as zero. For determining estimability with a nonzero x , $\beta'x$ is assessed by whether or not $\ N'x\ ^2 < \tau\ x'x\ ^2$, where N and τ denote <code>nbasis</code> and <code>tol</code> , respectively.
<code>rank</code>	Optional integer value. If not <code>NULL</code> , it is used in place of <code>tol</code> to decide the rank of the model matrix
<code>pivot</code>	Optional integer vector of length equal to the number of columns in the model matrix. If non- <code>NULL</code> , it is assumed that this specifies the order of columns in <code>x</code> relative to the order of predictors in the model matrix, and the returned non-estimable basis is permuted accordingly.
<code>nbasis</code>	Matrix whose columns span the null space of the model matrix. Such a matrix is returned by <code>nonest.basis</code> .

Details

Consider a linear model $y = X\beta + E$. If X is not of full rank, it is not possible to estimate β uniquely. However, $X\beta$ is uniquely estimable, and so is $a'X\beta$ for any conformable vector a . Since $a'X$ comprises a linear combination of the rows of X , it follows that we can estimate any linear function where the coefficients lie in the row space of X . Equivalently, we can check to ensure that the coefficients are orthogonal to the null space of X . Because estimability constraints are entirely determined by X , these methods apply to a rich class of models that includes linear models, generalized linear models, mixed models, and models with more general correlated-error structures (e.g., spatial models, time series models).

The `nonest.basis` method for class `'svd'` is not really functional as a method because there is no `"svd"` class (at least in R $\leq 4.2.0$). But the function `nonest.basis.svd` is exported and may be called directly; it works with results of [svd](#) or [La.svd](#). We *require* `x$v` to be the complete matrix of right singular values; but we do not need `x$u` at all.

The default method does serve as an `svd` method, in that it only works if `x` has the required elements of an SVD result, in which case it passes it to `nonest.basis.svd`.

The `matrix` method runs `nonest.basis.svd(svd(x, nu = 0))`. The `lm` method runs the `qr` method on `x$qr`.

The function `legacy.nonest.basis` is the original default method in early versions of the **estimability** package. It may be called with `x` being either a matrix or a `qr` object, and after obtaining the `R` matrix, it uses an additional QR decomposition of `t(R)` to obtain the needed basis. (The current `nonest.basis` method for `qr` objects is instead based on the singular-value decomposition of `R`, and requires much simpler code.)

The constant `all.estble` is simply a `1 x 1` matrix of `NA`. This specifies a trivial non-estimability basis, and using it as `nbasis` will cause everything to test as estimable.

Value

When X is not full-rank, the methods for `nonest.basis` return a basis for the null space of X . The number of rows is equal to the number of regression coefficients (*including* any NAs); and the number of columns is equal to the rank deficiency of the model matrix. The columns are orthonormal. If the model is full-rank, then `nonest.basis` returns `all.estble`. The `matrix` method uses X itself, the `qr` method uses the QR decomposition of X , and the `lm` method recovers the required information from the object.

The function `is.estble` returns a logical value (or vector, if `x` is a matrix) that is `TRUE` if the function is estimable and `FALSE` if not.

Use with Cholesky decompositions

If the `chol` function is called with `pivot = TRUE`, the returned matrix is suitable to provide to the `matrix` method of `nonest.basis`, and the returned basis is correctly pivoted without specifying `pivot` explicitly.

This works because the Cholesky decomposition R of $X'X$ (as constructed by `chol()`) satisfies $X'X = R'R$, implying that the rows of $X'X$ are linear combinations of the rows of R ; and they are both of equal rank - so R and $X'X$ have the same row space.

Examples

```
require(estimability)

X <- cbind(rep(1,5), 1:5, 5:1, 2:6)
( nb <- nonest.basis(X) )

# via the singular-value decomposition
SVD <- svd(X, nu = 0)    # we don't need the U part of UDV'
nonest.basis.svd(SVD)    # same result as above

# Via the Cholesky decomposition of X'X
ch <- chol(t(X) %*% X, pivot = TRUE) |> suppressWarnings()
( nb.chol <- nonest.basis(ch) )

# Test estimability of some linear functions for this X matrix
lfs <- rbind(c(1,4,2,5), c(2,3,9,5), c(1,2,2,1), c(0,1,-1,1))
is.estble(lfs, nb)
is.estble(lfs, nb.chol) # same results even though nb.chol is not identical to nb
```

```
# Illustration on 'lm' objects:
warp.lm1 <- lm(breaks ~ wool * tension, data = warpbreaks,
  subset = -(26:38),
  contrasts = list(wool = "contr.treatment", tension = "contr.treatment"))
zapsmall(warp.nb1 <- nonest.basis(warp.lm1))

# different parameterization of the same model:
warp.lm2 <- update(warp.lm1,
  contrasts = list(wool = "contr.sum", tension = "contr.helmert"))
zapsmall(warp.nb2 <- nonest.basis(warp.lm2))

# These bases depend of the parameterizations, but they both
# correctly identify the empty cell
wcells = with(warpbreaks, expand.grid(wool = levels(wool), tension = levels(tension)))
epredict(warp.lm1, newdata = wcells, nbasis = warp.nb1)
epredict(warp.lm2, newdata = wcells, nbasis = warp.nb2)
```

Index

- * **datasets**
 - nonest.basis, [7](#)
- * **models**
 - estimability-package, [2](#)
- * **package**
 - estimability-package, [2](#)
- * **regression**
 - estimability-package, [2](#)
- all.estble (nonest.basis), [7](#)
- chol, [9](#)
- epredict, [2](#), [3](#)
- estble.subspace, [2](#), [6](#)
- estimability (estimability-package), [2](#)
- estimability-package, [2](#)
- eupdate (epredict), [3](#)
- is.estble, [2](#), [4](#), [6](#)
- is.estble (nonest.basis), [7](#)
- La.svd, [8](#)
- legacy.nonest.basis (nonest.basis), [7](#)
- nonest.basis, [2](#), [4](#), [6](#), [7](#)
- predict, [2](#), [4](#)
- predict.lm, [4](#)
- svd, [8](#)
- update, [4](#)