Estimability Tools for Package Developers

by Russell V. Lenth

Abstract When a linear model is rank-deficient, then predictions based on that model become questionable because not all predictions are uniquely estimable. However, some of them are, and the **estimability** package provides tools that package developers can use to tell which is which. With the use of these tools, a model object's `predict` method could return estimable predictions as-is while flagging non-estimable ones in some way, so that the user can know which predictions to believe. The **estimability** package also provides, as a demonstration, an estimability-enhanced `epredict` method to use in place of `predict` for models fitted using the **stats** package.

Introduction

Consider a linear regression or mixed model having fixed component of the matrix form $\mathbf{X}\boldsymbol{\beta}$. If $\mathbf{X}$ is not of full column rank, then there is not a unique estimate $\mathbf{b}$ of $\boldsymbol{\beta}$. However, consider using $\boldsymbol{\lambda}'\mathbf{b}$ to estimate the value of some linear function $\boldsymbol{\lambda}'\boldsymbol{\beta} = \sum_j \lambda_j \beta_j$. (We use $\mathbf{x}'$ to denote the transpose of a vector $\mathbf{x}$.) For some $\boldsymbol{\lambda}$s, the prediction depends on the solution $\mathbf{b}$; but for others—the estimable ones—it does not.

An illustration

An example will help illustrate the issues. In the following commands, we create four predictors `x1`–`x4` and a response variable `y`:

```
> 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)
```

Clearly, `x3` and `x4` depend linearly on `x1` and `x2` and the intercept. Let us fit two versions of the same model to these data, entering the predictors in different orders, and compare the regression coefficients:

```
> mod1234 <- lm(y ~ x1 + x2 + x3 + x4)
> mod4321 <- lm(y ~ x4 + x3 + x2 + x1)
> zapsmall(rbind(b1234 = coef(mod1234), b4321 = coef(mod4321)[c(1, 5:2)]))
```

```
      (Intercept) x1 x2 x3 x4
b1234           5  3  0 NA NA
b4321         -19 NA NA  3  6
```

Note that in each model, two regression coefficients are `NA`. This indicates that the associated predictors were excluded due to their linear dependence on the others.

The problem that concerns us is making predictions on new values of the predictors. Here are some predictor combinations to try:

```
> 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))
```

And here is what happens when we make the predictions:

```
> cbind(testset,
+     pred1234 = predict(mod1234, newdata = testset),
+     pred4321 = suppressWarnings(predict(mod4321, newdata = testset)))
```

```
  x1 x2 x3 x4 pred1234 pred4321
1  3  1  7  2       14       14
2  6  2 14  4       23       47
```

```
3  6  2 14  0       23       23
4  0  0  0  4        5        5
5  0  0  0  0        5      -19
6  1  2  3  4        8       14

Warning message:
In predict.lm(mod1234, new = testset) :
  prediction from a rank-deficient fit may be misleading
```

Note that the two models produce identical predictions in some cases (rows 1, 3, and 4), and different ones in the others. There is also a warning message (we suppressed this the second time) telling us that this could happen.

It happens that cases 1, 3, and 4 are estimable, and the others are not. It would be helpful to know which predictions to trust, rather than a vague warning. The `epredict` function in **estimability** (Lenth, 2015) accomplishes this:

```
> require("estimability")
> rbind(epred1234 = epredict(mod1234, newdata = testset),
+       epred4321 = epredict(mod4321, newdata = testset))

           1  2  3 4  5  6
epred1234 14 NA 23 5 NA NA
epred4321 14 NA 23 5 NA NA
```

The results for non-estimable cases are indicated by `NA`s. Note that in both models, the same new-data cases are identified as estimable, and they are the ones that yield the same predictions. Note also that estimability was determined separately from each model. We do not need to compare two different fits to determine estimability.

Adding estimability checking to a modeling package

It is a simple matter to add estimability checking to the `predict` method(s) in a new or existing package.

1. The package should import the **estimability** package.
2. Use `estimability::nonest.basis` to obtain the basis for non-estimable functions of the regression coefficients, preferably in the model-fitting code.
3. In the `predict` method, call `estimability::is.estble` on the model matrix for new data. (It is not necessary to check estimability of predictions on the original data.)

This is implemented in code in a manner something like this:

```
fitmodel <- function(...) {
    ...
        code to create:
            'X' (the fixed-effects model matrix),
            'QR' (QR decomp of 'X'),
            'object' (the fitted model object)
    ...
    object$nonest <- estimability::nonest.basis(QR)
    object
}

predict.mymodel <- function(object, newdata, tol = 1e-8, ...) {
    ...
    if(!missing(newdata)) {
        ...
            code to create:
                'newX' (the model matrix for 'newdata')
        ...
        estble <- estimability::is.estble(newX, object$nonest, tol)

        ...
            code to flag non-estimable predictions
        ...
```

```
    }
    ...
}
```

The `nonest.basis` function returns a matrix whose columns span the null space of the model matrix **X**. If, for some reason, the QR decomposition is not used in model fitting, there is also a `nonest.basis` method that can be called with **X** itself. But if the QR decomposition of **X** is available, it is most efficient to use it.

The `is.estble` function tests each row of its first argument for estimability against the null basis provided in its second argument, and returns a logical vector. A third argument may be added to adjust the tolerance used in this test. It is important to remember that *all columns* of the model matrix be included. In particular, do not exclude the columns corresponding to `NA`s in the coefficient estimates, as they are needed for the estimability testing. Typically, the results of `is.estble` would be used to skip any non-estimable predictions and replace them with `NA`.

If there is not a rank deficiency in the model, `nonest.basis` returns `estimability::all.estble`, which is a trivial null basis (still of '`matrix`' class) with which `is.estble` will return all `TRUE`. The `nonest.basis` function, when called with a '`qr`' object, can immediately detect if there is no rank deficiency and will return `all.estble`. If your model-fitting algorithm does not use the QR decomposition, it may be worth including additional code to check for non-singular models, in which case it sets the null basis to `estimability::all.estble`, rather than have `nonest.basis` perform the additional computation to figure this out.

The sample code above is typical for the S3 object model. If S4 objects are being used, you may want to instead include a slot of class '`matrix`' for `nonest`; or incorporate `nonest` as part of some other slot.

To illustrate briefly, consider the previous example. The null basis for `mod1234` is obtained as follows:

```
> (nb <- nonest.basis(mod1234$qr))

            [,1]       [,2]
[1,]  0.29555933 -0.9176629
[2,]  0.76845426  0.2294157
[3,] -0.48767290 -0.2294157
[4,] -0.28078136  0.0000000
[5,] -0.07388983  0.2294157
```

and we can test estimability for the data in `testset` by converting it to a matrix and appending a column for the intercept:

```
> newX <- cbind(1, as.matrix(testset))
> is.estble(newX, nb)

[1]  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

Theory of estimability

The theory of estimability in linear models is well established, and can be found in almost any text on linear models, such as classics like Searle (1997) and Seber and Lee (2003). In this discussion, I will make specific references to portions of a more recent reference, Monahan (2008).

Before delving in, though, it is worth noting (based on what is said in some contributed packages' documentation and code) that there seems to be some confusion in the R community between non-estimability of a linear function $\boldsymbol{\lambda}'\boldsymbol{\beta}$ and the placement of nonzero λ_j coefficients relative to the positions of `NA` values in the estimate **b**. It is not possible to assess estimability with this information. Note, for instance, in the example in the Introduction, we obtained two different **b**s. Between them we can find an `NA` in the estimates for every predictor except the intercept. In fact, while `NA` usually refers to unknown values, that is not the case here. An `NA` regression coefficient signals a coefficient that is constrained to zero in order to obtain an estimate. And there is nothing wrong with multiplying some of the λ_j by zero. To assess estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$, we must look at *all* of the elements of $\boldsymbol{\lambda}$, even those corresponding to `NA`s in the estimates.

General results

When **X** is not full-rank, the solution **b** to the normal equations is not unique. However, the predicted values, **Xb**, are uniquely estimable; that is, for any two solutions $\mathbf{b}_1$ and $\mathbf{b}_2$, $\mathbf{X}\mathbf{b}_1 = \mathbf{X}\mathbf{b}_2$. Note that the

ith element of $\mathbf{Xb}$ is $\mathbf{x}_i'\mathbf{b}$ where $\mathbf{x}_i'$ is the ith row of $\mathbf{X}$. Thus, $\mathbf{x}_i'\boldsymbol{\beta}$ is estimable for each i, and so are any linear combinations of these. Indeed, $\boldsymbol{\lambda}'\boldsymbol{\beta}$ is estimable if and only if $\boldsymbol{\lambda}'$ is in the row space of $\mathbf{X}$—or equivalently, $\boldsymbol{\lambda}$ is in the column space of $\mathbf{X}'$ (Monahan, 2008, Result 3.1).

From the above result, we can establish estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$ either by showing that $\boldsymbol{\lambda}$ is in the column space of $\mathbf{X}'$, or by showing that it is orthogonal to the null space of $\mathbf{X}$ (Monahan, 2008, Methods 3.2 and 3.3, respectively). One way to implement the second idea is to note that if $(\mathbf{X}'\mathbf{X})^-$ is any generalized inverse of $\mathbf{X}'\mathbf{X}$, then $\mathbf{P} = (\mathbf{X}'\mathbf{X})(\mathbf{X}'\mathbf{X})^-$ is a projection onto the column space of $\mathbf{X}'$, and so $\mathbf{I} - \mathbf{P}$ is a projection onto the null space of $\mathbf{X}$. The columns of $\mathbf{I} - \mathbf{P}$ thus comprise a basis for this null space. Accordingly, estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$ can be determined by showing that $\boldsymbol{\lambda}'(\mathbf{I} - \mathbf{P}) = \mathbf{0}'$.

SAS uses $\mathbf{I} - \mathbf{P}$, as described above, to test estimability. Of course, in computation we need to set a tolerance for being close enough to $\mathbf{0}$. SAS deems $\boldsymbol{\lambda}'\boldsymbol{\beta}$ non-estimable if $\boldsymbol{\lambda} \neq \mathbf{0}$ and $\max|\boldsymbol{\lambda}'(\mathbf{I} - \mathbf{P})| > \psi \cdot \max|\boldsymbol{\lambda}|$, where ψ is a tolerance factor with a default value of 10^{-4}. For details, see SAS Institute Inc. (2013), the chapter on "Shared Concepts," documentation for the `ESTIMATE` statement and its `SINGULAR` option.

Methods used by estimability

In the **estimability** package, we obtain the null basis in a different way than shown above, in part because the QR decomposition of $\mathbf{X}$ is already (usually) available in an '`lm`' object. Suppose that $\mathbf{X}$ is $n \times p$, then we can write $\mathbf{X} = \mathbf{QR}$, where $\mathbf{Q}$ ($n \times p$) has orthonormal columns, and $\mathbf{R}$ ($p \times p$) is upper triangular with nonzero diagonal elements. If $\mathbf{X}$ is of rank $r < p$, then columns are pivoted so that the linearly dependent ones come last, and we only need r columns of $\mathbf{Q}$ and the top r rows of $\mathbf{R}$, along with the pivoting information. Call these pivoted, down-sized matrices $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{R}}$ respectively; and let $\widetilde{\mathbf{X}}$ (still $n \times p$) denote $\mathbf{X}$ with its columns permuted according to the pivoting. We now have that $\widetilde{\mathbf{X}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$, and $\widetilde{\mathbf{R}}$ comprises the first r rows of an upper triangular matrix. Observe that each row of $\widetilde{\mathbf{X}}$ is thus a linear combination of the rows of $\widetilde{\mathbf{R}}$—that is, the row space of $\widetilde{\mathbf{X}}$ is the same as the row space of $\widetilde{\mathbf{R}}$. So the much smaller matrix $\widetilde{\mathbf{R}}$ has everything we need to know to determine estimability.

Now, let $d = p - r$ denote the rank deficiency, and define

$$\mathbf{S} = \left[\begin{array}{c|c} \widetilde{\mathbf{R}}'_{p\times r} & \begin{array}{c} \mathbf{0}_{r\times d} \\ \mathbf{I}_{d\times d} \end{array} \end{array}\right]$$

Then $\mathbf{S}$ is an upper triangular matrix with nonzero diagonal—hence it is of full rank $p = r + d$. Let $\mathbf{S} = \mathbf{TU}$ be the QR decomposition of $\mathbf{S}$. We then have that $\mathbf{T}$ has orthonormal columns, and in fact it is a Gram-Schmidt orthonormalization of $\mathbf{S}$. Accordingly, the first r columns of $\mathbf{T}$ comprise an orthonormalization of the columns of $\widetilde{\mathbf{R}}'$, and thus they form a basis for the row space of $\widetilde{\mathbf{R}}$ and hence of $\widetilde{\mathbf{X}}$. Letting $\widetilde{\mathbf{N}}$ denote the last d columns of $\mathbf{T}$, we have that $\widetilde{\mathbf{N}}$ has orthonormal columns, all of which are orthogonal to the first r columns of $\mathbf{T}$ and hence to the row space of $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{X}}$. It follows that $\widetilde{\mathbf{N}}$ is an orthonormal basis for the null space of $\widetilde{\mathbf{X}}$. After permuting the rows of $\widetilde{\mathbf{N}}$ according to the original pivoting in the QR decomposition of $\mathbf{X}$, we obtain a basis $\mathbf{N}$ for the null space of $\mathbf{X}$.

To test estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$, first compute $\mathbf{z}' = \boldsymbol{\lambda}'\mathbf{N}$, which is a d-vector. Theoretically, $\boldsymbol{\lambda}'\boldsymbol{\beta}$ is estimable if and only if $\mathbf{z}'\mathbf{z} = \mathbf{0}$; but for computational purposes, we need to set a tolerance for its being suitably small. Again, we opt to deviate from SAS's approach, which is based on $\max|z_i|$. Instead, we deem $\boldsymbol{\lambda}'\boldsymbol{\beta}$ non-estimable when $\boldsymbol{\lambda} \neq \mathbf{0}$ and $\mathbf{z}'\mathbf{z} \geq \tau \cdot \boldsymbol{\lambda}'\boldsymbol{\lambda}$, where τ is a tolerance value. Our default value is $\tau = (10^{-4})^2 = 10^{-8}$. The rationale for this criterion is that it is rotation-invariant: We could replace $\mathbf{N} \leftarrow \mathbf{VN}$ where $\mathbf{V}$ is any $p \times p$ orthogonal matrix, and $\mathbf{z}'\mathbf{z}$ will be unchanged. Such rotation invariance seems a desirable property because the assessment of estimability does not depend on which null basis is used.

Bells and whistles

The **estimability** package's `epredict` function serves as a demonstration of adding estimability checking to the `predict` methods in the **stats** package. It works for `lm`, `aov`, `glm`, and `mlm` objects. It also provides additional options for the `type` argument of `predict`. When `newdata` is provided, `type = "estimability"` returns a logical vector showing which rows of `newdata` are estimable; and `type = "matrix"` returns the model matrix for `newdata`.

An accompanying enhancement is `eupdate`, which runs `update` on a model object and also adds a `nonest` member. Subsequent `epredict` calls on this object will use that as the null basis instead of having to reconstruct it. For example,

```
> mod123 <- eupdate(mod1234, . ~ . - x4)
```

```
> mod123$nonest

           [,1]
[1,]  0.0000000
[2,] -0.8017837
[3,]  0.5345225
[4,]  0.2672612
```

Now let's find out which predictions in `testset` are estimable with this model:

```
> epredict(mod123, newdata = testset, type = "estimability")

    1     2     3     4     5     6
 TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

Thus, more of these test cases are estimable when we drop x_4 from the model. Looking at `testset`, this makes sense because two of the previously non-estimable rows differ from estimable ones only in their values of x_4.

Conclusion

In rank-deficient linear models used for predictions on new predictor sets, it is important for users to know which predictions are to be trusted, and which of them would change if the same model had been specified in a different way. The **estimability** package provides an easy way to add this capability to new and existing model-fitting packages.

Bibliography

R. V. Lenth. *estimability: Estimability Tools for Linear Models*, 2015. URL http://CRAN.R-project.org/package=estimability. R package version 1.1. [p196]

J. F. Monahan. *A Primer on Linear Models*. Chapman & Hall/CRC, 2008. [p197, 198]

SAS Institute Inc. *SAS/STAT Software, Version 13.2*. SAS Institute Inc., Cary, NC, 2013. URL http://www.sas.com/. [p198]

S. R. Searle. *Linear Models*. Wiley Classics. Wiley-Interscience, 1997. [p197]

G. A. F. Seber and A. J. Lee. *Linear Regression Analysis*. John Wiley & Sons, Inc., second edition, 2003. [p197]

Russell V. Lenth
The University of Iowa
241 Schaeffer Hall
Iowa City, IA 52242
USA
russell-lenth@uiowa.edu