Identification and estimable functions

Simen Gaure

ABSTRACT. A walkthrough of the identification problems which may arise in models with many dummies, and how **lfe** handles them.

1

The **lfe** package is used for ordinary least squares estimation, i.e. models which conceptually may be estimated by `lm` as

```
> lm(y ~ x1 + x2 + ... + f1 + f2 + ... + fn)
```

where `f1,f2,...,fn` are factors. The standard method is to introduce a dummy variable for each level of each factor. This is too much as it introduces multicollinearities in the system. Conceptually, the system may still be solved, but there are many different solutions. In all of them, the difference between the coefficients for each factor will be the same.

The ambiguity is typically solved by removing a single dummy variable for each factor, this is termed a reference. This is like forcing the coefficient for this dummy variable to zero, and the other levels are then seen as relative to this zero. Other ways to solve the problem is to force the sum of the coefficients to be zero, or one may enforce some other constraint, typically via the `contrasts` argument to `lm`. The default in `lm` is to have a reference level in each factor, and a common intercept term.

In **lfe** the same estimation can be performed by

```
> felm(y ~ x1 + x2 + ... + G(f1) + G(f2) + ... + G(fn))
```

Since `felm` conceptually does exactly the same as `lm`, the contrasts approach may work there too. Or rather, it is actually not necessary that `felm` handles it at all, it is only necessary if one needs to fetch the coefficients for the factor levels with `getfe`.

lfe is intended for very large datasets, with factors with many levels. Then the approach with a single constraint for each factor may sometimes not be sufficient. The standard example in the econometrics literature is the case with two factors, one for individuals, and one for firms these individuals work for, changing jobs now and then. What happens in practice is that the labour market may be disconnected, so that one set of individuals move between one set of firms, and another (disjoint) set of individuals move between some other firms. This happens for no obvious reason, and is data dependent, not intrinsic to the model. There may be several such components. I.e. there are more multicollinearities in the system than the obvious ones. In such a case, there is no way to compare coefficients from different connected components, it is not sufficient with a single individual reference. The problem may be phrased in graph theoretic terms, and it can be shown that it is sufficient with one reference level in each of the connected components. This is what **lfe** does, in the case with two factors it identifies these components, and force one level to zero in one of the factors.

2. Identification with two factors

In the case with two factors, i.e. two `G()` terms in the model, identification is well-known. `getfe` will partition the dataset into connected components, and introduce a reference level in each component:

```
> library(lfe)
> set.seed(42)
> x1 <- rnorm(20)
> f1 <- sample(8,length(x1),replace=TRUE)/10
> f2 <- sample(8,length(x1),replace=TRUE)/10
> e1 <- sin(f1) + 0.02*f2^2 + rnorm(length(x1))
> y <-  2.5*x1 + (e1-mean(e1))
> summary(est <- felm(y ~ x1 + G(f1) + G(f2)))
Call:
   felm(formula = y ~ x1 + G(f1) + G(f2))

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3993 -0.2794  0.0000  0.4362  0.9813 

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   2.5305     0.3771    6.71  0.00111 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.126 on 5 degrees of freedom
Multiple R-squared: 0.9735   Adjusted R-squared: 0.8938 
F-statistic:  13.1 on 14 and 5 DF, p-value: 0.005105
```

We examine the estimable function produced by `efactory`.

```
> ef <- efactory(est,'ref')
> is.estimable(ef,est$fe)
[1] TRUE
> getfe(est)
             effect obs comp fe idx
f1.0.1  0.376275188   2    1 f1 0.1
f1.0.2 -0.081099976   1    2 f1 0.2
f1.0.3 -0.686880307   3    1 f1 0.3
f1.0.4  0.573177496   4    1 f1 0.4
f1.0.5  0.479141881   2    1 f1 0.5
f1.0.6  1.413019541   3    1 f1 0.6
```

```
f1.0.7  0.844955931   1    2 f1 0.7
f1.0.8  0.926433813   4    1 f1 0.8
f2.0.1 -0.004011328   3    1 f2 0.1
f2.0.2  0.000000000   5    1 f2 0.2
f2.0.3 -1.518666584   1    1 f2 0.3
f2.0.4  0.000000000   2    2 f2 0.4
f2.0.5 -1.894523687   2    1 f2 0.5
f2.0.6 -0.884319222   3    1 f2 0.6
f2.0.7 -0.609110270   3    1 f2 0.7
f2.0.8 -0.968652469   1    1 f2 0.8
```

As we can see from the `comp` entry, there are two components, with `f1=0.2`, `f1=0.7` and `f2=0.4`. A reference is introduced in each of the components, i.e. `f2.0.2=0` and `f2.0.4=0`. If we look at the dataset, the component structure becomes clearer:

```
> data.frame(f1,f2,comp=est$cfactor)
    f1  f2 comp
1  0.4 0.6    1
2  0.4 0.8    1
3  0.1 0.7    1
4  0.8 0.5    1
5  0.4 0.7    1
6  0.8 0.2    1
7  0.8 0.3    1
8  0.6 0.7    1
9  0.8 0.6    1
10 0.5 0.2    1
11 0.3 0.1    1
12 0.3 0.2    1
13 0.4 0.2    1
14 0.7 0.4    2
15 0.1 0.2    1
16 0.6 0.6    1
17 0.6 0.1    1
18 0.2 0.4    2
19 0.3 0.5    1
20 0.5 0.1    1
```

Observation 14 and 18 belong to component 2; no other observation has `f1=0.7`, `f1=0.2` or `f2=0.4`, thus it is clear that coefficients for these can not be compared to other coefficients.

3. Identification with three or more factors

In the case with three or more factors, there is no general intuitive theory (yet) for handling identification problems. **lfe** resorts to the simple-minded approach that non-obvious multicollinearities arise among the first two factors, and assumes it is sufficient with a single reference level for each of the remaining factors. In other words, the order of the factors in the model specification is important. A typical example would be 3 factors; individuals, firms and education:

```
> est <- felm(logwage ~ x1 + x2 + G(id) + G(firm) + G(edu))
> getfe(est)
```

This will result in exactly the same references as if using the model

```
> logwage ~ x1 + x2 + G(id) + G(firm) + edu
```

though it may run faster (or slower).

Alternatively, one could specify the model as

```
> logwage ~ x1 + x2 + G(firm) + G(edu) + G(id)
```

This would not account for a partioning of the labour market along individual/firm, but along firm/education, using a single reference level for the individuals. In this example, there is some reason to suspect that it is not sufficient, depending on how `edu` is specified. There exists no general scheme that sets up suitable reference groups when there are more than two factors. It may happen that the default is sufficient. The function `getfe` will check whether this is so, and it will yield a warning about 'non-estimable function' if not. With some luck it may be possible to rearrange the order of the factors to avoid this situation.

There is nothing special with **lfe** in this respect. You will meet the same problem with `lm`, it will remove a reference level (or dummy-variable) in each factor, but the system will still contain multicollinearities. You may remove reference levels until all the multicollinearities are gone, but there is no obvious way to interpret the resulting coefficients.

To illustrate, the classical example is when you include a factor for age (in years), a factor for observation year, and a factor for year of birth. You pick a reference individual, e.g. `age=50`, `year=2013` and `birth=1963`, but this is not sufficient to remove all the multicollinearities. If you analyze this problem you will find that the coefficients are only identified up to linear trends. You may force the linear trend between `birth=1963` and `birth=1990` to zero, by removing the reference level `birth=1990`, and the system will be free of multicollinearities. In this case the `birth` coefficients have the interpretation as being deviations from a linear trend, though you do not know which linear trend. The `age` and `year` coefficients are also relative to this unknown trend in the `birth`-coefficients.

In the above case, the multicollinearity is obviously built into the model, and it is possible to remove it and find some intuitive interpretation of the coefficients. In the general case, when either `lm` or `getfe` reports a handful of non-obvious spurious multicollinearites between factors with many levels, you probably will not be able to find any reasonable way to interpret coefficients. Of course, certain linear combinations of coefficients will be unique, i.e. estimable, and for small datasets these may be found by e.g. the algorithm in [**1**], but the general picture is muddy.

lfe does not provide a solution to this problem, however, `getfe` will still provide a vector of coefficients which results from finding a non-unique solution to a certain set of equations. To get any sense from this, an estimable function must be applied. The simplest one is to pick a reference for each factor and subtract this coefficient from each of the other coefficients in the same factor, and add it to a common intercept, however in the case this does not result in an estimable function, you are out of luck. If you for some reason believe that you know of an estimable function, you may provide this to `getfe` via the `ef`-argument. There is an example in the `getfe` documentation. You may also test it for estimability with the function `is.estimable`, this is a probabilistic test which almost never fails.

4. Specifying an estimable function

A model of the type

```
> y ~ x1 + x2 + f1 + f2 + f3
```

may be written in matrix notation as

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

where X is a matrix with columns `x1` and `x2` and D is matix of dummies constructed from the levels of the factors `f1,f2,f3`. Formally, an estimable function in our context is a matrix operator whose row space is contained in the row space of D. That is, an estimable function may be written as a matrix. Like the `contrasts` argument to `lm`. However, the **lfe** package uses an R-function instead. That is, `felm` is called first:

```
> est <- felm(y ~ x1 + x2 + G(f1)+G(f2)+G(f3))
```

This yields the parameters for `x1` and `x2`, i.e. $\hat{\beta}$. To find the parameters for the levels of `f1,f2,f3`, a certain linear system is solved:

$$D\gamma = R \tag{1}$$

where R can be computed when we have β. This does not identify γ uniquely, we have to apply an estimable function to γ. The estimable function F is characterized by the property that $F\gamma_1 = F\gamma_2$ whenever γ_1 and γ_2 are solutions to equation (1). Rather than coding F as a matrix, **lfe** codes it as a function. It is of course possible to let the function apply a matrix, so this is not a material distinction. So, let's look at an example of how an estimable function may be made:

```
> library(lfe)
> x1 <- rnorm(100)
> f1 <- sample(7,100,replace=TRUE)
> f2 <- sample(8,100,replace=TRUE)/8
> f3 <- sample(10,100,replace=TRUE)/10
> e1 <- sin(f1) + 0.02*f2^2  + 0.17*f3^3 + rnorm(100)
> y <-  2.5*x1 + (e1-mean(e1))
> summary(est <- felm(y ~ x1 + G(f1) + G(f2) + G(f3)))
Call:
   felm(formula = y ~ x1 + G(f1) + G(f2) + G(f3))

Residuals:
      Min        1Q    Median        3Q       Max
-2.740510 -0.567337 -0.007152  0.634233  2.023829

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   2.4885     0.1308   19.03   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.027 on 76 degrees of freedom
Multiple R-squared: 0.8557   Adjusted R-squared: 0.8102
```

```
F-statistic:  19.6 on 23 and 76 DF, p-value: < 2.2e-16
*** Standard errors may be slightly too high due to more than 2 groups
```

In this case, with 3 factors we can not be certain that it is sufficient with a single reference in two of the factors, but we try it as an exercise. (**lfe** does not include an intercept, it is subsumed in one of the factors, so it should tentatively be sufficient with a reference for the two others).

The input to our estimable function is a solution γ of equation (1). The argument `addnames` is a logical, set to `TRUE` when the function should add names to the resulting vector. The coefficients is ordered the same was as the levels in the factors. We should pick a single reference in factors `f2,f3`, subtract these, and add the sum to the first factor:

```
> ef <- function(gamma,addnames) {
+   ref2 <- gamma[[8]]
+   ref3 <- gamma[[16]]
+   gamma[1:7] <- gamma[1:7]+ref2+ref3
+   gamma[8:15] <- gamma[8:15]-ref2
+   gamma[16:25] <- gamma[16:25]-ref3
+   if(addnames) {
+     names(gamma) <- c(paste('f1',1:7,sep='.'),
+                       paste('f2',1:8,sep='.'),
+                       paste('f3',1:10,sep='.'))
+   }
+   gamma
+ }
> is.estimable(ef,fe=est$fe)
```

```
[1] TRUE
```

```
> getfe(est,ef=ef)
```

```
          effect
f1.1   0.87192373
f1.2   0.88686743
f1.3  -0.46439446
f1.4  -0.63725214
f1.5  -1.25719035
f1.6  -0.35436953
f1.7   0.80583446
f2.1   0.00000000
f2.2  -0.13071989
f2.3   0.15364412
f2.4   0.16115681
f2.5   0.17442275
f2.6   0.34237855
f2.7   0.55430414
f2.8   0.66755401
f3.1   0.00000000
f3.2  -0.24234222
f3.3  -0.44140183
f3.4  -0.39577038
```

```
f3.5  -0.39789618
f3.6   0.05501249
f3.7   0.47014646
f3.8  -0.46291012
f3.9   0.08411682
f3.10 -0.16182575
```

We may compare this to the default estimable function, which picks a reference in each connected component as defined by the two first factors.

```
> getfe(est)
```

```
               effect obs comp fe   idx
f1.1      0.000000000  20    1 f1     1
f1.2      0.014943693  12    1 f1     2
f1.3     -1.336318194  14    1 f1     3
f1.4     -1.509175872  16    1 f1     4
f1.5     -2.129114082  13    1 f1     5
f1.6     -1.226293262  11    1 f1     6
f1.7     -0.066089281  14    1 f1     7
f2.0.125  0.474027558  14    1 f2 0.125
f2.0.25   0.343307662  20    1 f2  0.25
f2.0.375  0.627671675  13    1 f2 0.375
f2.0.5    0.635184365   6    1 f2   0.5
f2.0.625  0.648450310  11    1 f2 0.625
f2.0.75   0.816406109  11    1 f2  0.75
f2.0.875  1.028331702  12    1 f2 0.875
f2.1      1.141581571  13    1 f2     1
f3.0.1    0.397896188   3    2 f3   0.1
f3.0.2    0.155553961   7    2 f3   0.2
f3.0.3   -0.043505649  15    2 f3   0.3
f3.0.4    0.002125793  12    2 f3   0.4
f3.0.5    0.000000000  19    2 f3   0.5
f3.0.6    0.452908664  11    2 f3   0.6
f3.0.7    0.868042633   4    2 f3   0.7
f3.0.8   -0.065013933  11    2 f3   0.8
f3.0.9    0.482013006   9    2 f3   0.9
f3.1      0.236070435   9    2 f3     1
```

We see that the default has some more information. It uses the level names, and some more information, added like this:

```
> efactory(est,'ref')
```

```
function (v, addnames)
{
    esum <- sum(v[extrarefs])
    df <- v[refsubs]
    sub <- ifelse(is.na(df), 0, df)
    df <- v[refsuba]
    add <- ifelse(is.na(df), 0, df + esum)
    v <- v - sub + add
    if (addnames) {
        names(v) <- nm
        attr(v, "extra") <- list(obs = obs, comp = comp, fe = fef,
            idx = idx)
    }
    v
}
<bytecode: 0x5b72468>
<environment: 0x63a6dd8>
```

I.e. when asked to provide level names, it is also possible to add additional information as a `list` (or `data.frame`) as an attribute `'extra'`. The vectors `extrarefs,refsubs,refsuba` etc. are precomputed by `efactory` for speed efficiency.

5. Non-estimability

Consider another example. To ensure spurious relations there are almost as many factor levels as there are observations, and it will be hard to find enough estimable function to interpret all the coefficients. The coefficient for `x1` is still estimated, but with a large standard error.

```
> set.seed(42)
> x1 <- rnorm(100)
> f1 <- sample(34,100,replace=TRUE)
> f2 <- sample(34,100,replace=TRUE)/8
> f3 <- sample(34,100,replace=TRUE)/10
> e1 <- sin(f1) + 0.02*f2^2  + 0.17*f3^3 + rnorm(100)
> y <-  2.5*x1 + (e1-mean(e1))
> summary(est <- felm(y ~ x1 + G(f1) + G(f2) + G(f3)))
```

```
Call:
   felm(formula = y ~ x1 + G(f1) + G(f2) + G(f3))

Residuals:
       Min         1Q     Median         3Q        Max 
-8.690e-01 -9.853e-02 -9.920e-12  1.135e-01  8.690e-01 

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   1.6543     0.8971   1.844    0.206

Residual standard error: 1.615 on 2 degrees of freedom
Multiple R-squared: 0.9958   Adjusted R-squared: 0.7906 
F-statistic: 4.903 on 97 and 2 DF, p-value: 0.1841 
*** Standard errors may be slightly too high due to more than 2 groups
```

The default estimable function fails, and the coefficients from `getfe` are not useable. `getfe` yields a warning in this case.

```
> ef <- efactory(est,'ref')
> is.estimable(ef,est$fe)
```

```
[1] FALSE
```

Indeed, the rank-deficiency is quite large. There are more spurious relations between the factors than what can be accounted for by looking at components in the two first factors. In this low-dimensional example we may find the matrix D of equation (1), and its rank which is lower than the number of columns:

```
> f1 <- factor(f1); f2 <- factor(f2); f3 <- factor(f3)
> D <- t(rBind(as(f1,'sparseMatrix'),
+              as(f2,'sparseMatrix'),
+              as(f3,'sparseMatrix')))
> dim(D)
[1] 100  99
> as.numeric(rankMatrix(D))
[1] 92
```

We can get an idea what happens if we keep the dummies for `f1`. In this case, with 2 factors, **lfe** will partition the dataset into connected components and account for all the multicollinearities among the factors `f2` and `f3`, but this is not sufficient. The interpretation of the resulting coefficients is not straightforward.

```
> summary(est <- felm(y ~ x1 + G(f2) + G(f3) + f1))
Call:
   felm(formula = y ~ x1 + G(f2) + G(f3) + f1)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.86895 -0.09853  0.00000  0.11346  0.86895 

Coefficients:
     Estimate Std. Error t value Pr(>|t|)  
x1    1.65426    0.47951   3.450   0.0107 *
f12  -1.90505    4.89449  -0.389   0.7087  
f13  -0.50215    1.82413  -0.275   0.7910  
f14  -6.22264    3.01472  -2.064   0.0779 .
f15  -3.25066    1.30713  -2.487   0.0418 *
f16  -0.90207    1.43495  -0.629   0.5495  
f17  -1.94779    2.31183  -0.843   0.4273  
f18   1.06828    2.19941   0.486   0.6420  
f19  -3.71630    1.74689  -2.127   0.0709 .
f110       NA         NA      NA       NA  
f111 -2.79296    2.03317  -1.374   0.2119  
f112 -2.39955    1.22205  -1.964   0.0903 .
f113       NA         NA      NA       NA  
f114  2.26528    1.84794   1.226   0.2599  
f115  0.50911    2.17930   0.234   0.8220  
f116  0.77581    1.84701   0.420   0.6871  
f117 -1.73116    1.45181  -1.192   0.2719  
f118       NA         NA      NA       NA  
f119 -0.10752    1.42174  -0.076   0.9418  
f120 -1.78120    1.96692  -0.906   0.3953  
f121  2.40789    1.95402   1.232   0.2576  
```

```
f122  2.96339    2.66996   1.110   0.3037
f123 -4.51110    5.50755  -0.819   0.4397
f125 -3.10254    2.41876  -1.283   0.2404
f126       NA         NA      NA       NA
f127 -0.98631    2.89668  -0.340   0.7435
f128 -0.54472    1.98226  -0.275   0.7914
f129  1.10020    2.85622   0.385   0.7115
f130 -4.42386    2.01494  -2.196   0.0642 .
f131 -0.31554    1.40158  -0.225   0.8283
f132  1.67510    1.87694   0.892   0.4018
f133 -0.04469    1.58114  -0.028   0.9782
f134  0.23692    2.21817   0.107   0.9179
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8633 on 7 degrees of freedom
Multiple R-squared: 0.9958   Adjusted R-squared: 0.9402
F-statistic: 18.09 on 92 and 7 DF, p-value: 0.0002557
```

```
> getfe(est)
```

```
                 effect obs comp fe   idx
f2.0.125 -0.3896981   1    1 f2 0.125
f2.0.25  -0.6084812   3    1 f2  0.25
f2.0.375 -2.5763220   4    1 f2 0.375
f2.0.5    1.9753019   7    1 f2   0.5
f2.0.625 -2.6204281   1    1 f2 0.625
f2.0.75   0.3344278   4    1 f2  0.75
f2.0.875  0.0000000   1    2 f2 0.875
f2.1     -0.8790835   2    1 f2     1
f2.1.25   0.2587411   5    1 f2  1.25
f2.1.375  3.7993936   5    1 f2 1.375
f2.1.5    0.3350056   1    1 f2   1.5
f2.1.625  0.4889111   4    1 f2 1.625
f2.1.75  -0.9947498   5    1 f2  1.75
f2.1.875  3.2224805   2    1 f2 1.875
f2.2     -4.0989311   3    1 f2     2
f2.2.125  1.6395098   6    1 f2 2.125
f2.2.25   3.1212805   1    1 f2  2.25
f2.2.375  3.0419158   2    1 f2 2.375
f2.2.5    3.1781146   5    1 f2   2.5
f2.2.625  4.1143538   2    1 f2 2.625
f2.2.75   1.8330435   1    1 f2  2.75
f2.2.875  0.3258495   4    1 f2 2.875
f2.3      7.1934760   1    1 f2     3
f2.3.125  1.3735941   3    1 f2 3.125
f2.3.25   1.8938729   3    1 f2  3.25
f2.3.5   -0.7071215   4    1 f2   3.5
f2.3.625  1.5205296   2    1 f2 3.625
f2.3.75   1.7182287   2    1 f2  3.75
```

```
f2.3.875 -3.5165466   3    1 f2 3.875
f2.4       0.5024135   5    1 f2     4
f2.4.125   0.2211940   5    1 f2 4.125
f2.4.25    0.1436990   3    1 f2  4.25
f3.0.1    -3.2713276   3    1 f3   0.1
f3.0.2     4.8662353   2    1 f3   0.2
f3.0.3     0.0000000   8    1 f3   0.3
f3.0.4    -4.0156531   4    1 f3   0.4
f3.0.5    -1.5600761   4    1 f3   0.5
f3.0.6    -1.8723661   3    1 f3   0.6
f3.0.7     1.9569422   2    1 f3   0.7
f3.0.8    -3.4556601   2    1 f3   0.8
f3.0.9    -1.9322916   4    1 f3   0.9
f3.1      -0.9823675   1    1 f3     1
f3.1.1    -2.0101596   4    1 f3   1.1
f3.1.2    -2.4877797   2    1 f3   1.2
f3.1.3     0.2322515   1    1 f3   1.3
f3.1.4    -1.5634550   5    1 f3   1.4
f3.1.5    -1.6201727   5    1 f3   1.5
f3.1.6     0.7950336   5    1 f3   1.6
f3.1.7    -0.5123151   5    1 f3   1.7
f3.1.8     2.4253869   3    1 f3   1.8
f3.1.9    -0.4532778   2    1 f3   1.9
f3.2       6.5804358   2    1 f3     2
f3.2.1     3.6123451   1    2 f3   2.1
f3.2.2     0.1686157   2    1 f3   2.2
f3.2.3     1.8276048   4    1 f3   2.3
f3.2.4     4.1913861   2    1 f3   2.4
f3.2.5    -2.3235774   1    1 f3   2.5
f3.2.6     2.5043263   1    1 f3   2.6
f3.2.7    -0.2265028   2    1 f3   2.7
f3.2.8     4.4396033   2    1 f3   2.8
f3.2.9     3.9948149   4    1 f3   2.9
f3.3       4.1069684   3    1 f3     3
f3.3.1     0.9933567   5    1 f3   3.1
f3.3.2     4.7385688   2    1 f3   3.2
f3.3.3     4.0585892   1    1 f3   3.3
f3.3.4     8.0138794   3    1 f3   3.4
```

References

[1] Godolphin, J.D., Godolphin, E.J., 2001. On the connectivity of row-column designs. Util. Math. 60, 51–65.

Ragnar Frisch Centre for Economic Research, Oslo, Norway