Multicollinearity, identification, and estimable functions

Simen Gaure

Abstract. Since there is quite a lot of confusion here and there about what happens when factors are collinear; here is a walkthrough of the identification problems which may arise in models with many dummies, and how **lfe** handles them. (Or, at the very least, attempts to handle them).

1. Context

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 (see e.g. [**2**]) 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 (see e.g. [**1, 3, 4**]), 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.

In the examples below, rather small randomly generated datasets are used. **lfe** is hardly the best solution for these problems, they are solely used to illustrate some concepts. I can assure the reader that no CPUs, sleeping patterns, romantic relationships, trees or cats, nor animals in general, were harmed during data collection and analysis.

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)
> is.estimable(ef,est$fe)
[1] TRUE
> getfe(est)
```

```
             effect obs comp fe idx
f1.0.1  0.376275185   2    1 f1 0.1
f1.0.2 -0.081099976   1    2 f1 0.2
f1.0.3 -0.686880302   3    1 f1 0.3
f1.0.4  0.573177494   4    1 f1 0.4
f1.0.5  0.479141883   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  0.926433817   4    1 f1 0.8
f2.0.1 -0.004011331   3    1 f2 0.1
f2.0.2  0.000000000   5    1 f2 0.2
f2.0.3 -1.518666585   1    1 f2 0.3
f2.0.4  0.000000000   2    2 f2 0.4
f2.0.5 -1.894523692   2    1 f2 0.5
f2.0.6 -0.884319224   3    1 f2 0.6
f2.0.7 -0.609110269   3    1 f2 0.7
f2.0.8 -0.968652465   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. `lm` is silent about this component structure, hence coefficients are hard to interpret. Though, predictive properties and residuals are the same:

```
> f1 <- factor(f1); f2 <- factor(f2)
> summary(lm(y ~ x1 + f1 + f2))
Call:
lm(formula = y ~ x1 + f1 + f2)

Residuals:
         1          2          3          4          5          6          7
 4.181e-01  4.718e-16  6.357e-01  4.906e-01 -1.399e+00 -2.261e-01  1.943e-16
         8          9         10         11         12         13         14
 7.637e-01 -2.645e-01  2.048e-01  8.148e-01 -3.243e-01  9.813e-01  1.527e-16
        15         16         17         18         19         20
-6.357e-01 -1.536e-01 -6.101e-01 -1.249e-16 -4.906e-01 -2.048e-01

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.372264   1.212957   0.307  0.77129
x1           2.530517   0.377104   6.710  0.00111 **
f10.2       -0.453364   2.028662  -0.223  0.83201
f10.3       -1.063155   1.248670  -0.851  0.43340
f10.4        0.196902   1.074944   0.183  0.86186
f10.5        0.102867   1.270095   0.081  0.93859
f10.6        1.036744   1.153771   0.899  0.41006
f10.7        0.472692   1.670068   0.283  0.78849
f10.8        0.550159   1.259446   0.437  0.68046
f20.2        0.004011   0.969131   0.004  0.99686
f20.3       -1.514655   1.563449  -0.969  0.37714
f20.4              NA         NA      NA       NA
f20.5       -1.890512   1.504645  -1.256  0.26446
f20.6       -0.880308   1.111697  -0.792  0.46434
f20.7       -0.605099   1.097276  -0.551  0.60506
f20.8       -0.964641   1.585714  -0.608  0.56954
---
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:  0.8991
F-statistic:  13.1 on 14 and 5 DF,  p-value: 0.005105
```

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, i.e. that they in principle could be specified as ordinary dummies. 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 the same number of 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 (see e.g. [**6**]) 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 between 1963 and 1990, though you do not know which linear trend. The `age` and `year` coefficients are also relative to this same unknown trend.

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 these may be found by e.g. the procedures in [**5, 8**], 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 (see [**4**, Remark 6.2]).

4. Specifying an estimable function

A model of the type

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

may be written in matrix notation as

$$y = X\beta + D\alpha + \epsilon, \tag{1}$$

where X is a matrix with columns `x1` and `x2` and D is matrix 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, it uses the Frisch-Waugh-Lovell theorem to project out the $D\alpha$ term from (1) (see [**4**, Remark 3.2]):

```
> 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 $\hat{\alpha}$, the parameters for the levels of `f1,f2,f3`, `getfe` solves a certain linear system (see [**4**, eq. (14)]):

$$D\gamma = \rho \tag{2}$$

where the vector ρ can be computed when we have $\hat{\beta}$. 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 (2). 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
-1.88686 -0.72519 -0.07878  0.75584  2.30499

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1    2.354      0.112   21.03   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.076 on 76 degrees of freedom
Multiple R-squared: 0.9005   Adjusted R-squared: 0.8691
```

```
F-statistic: 29.91 on 23 and 76 DF, p-value: < 2.2e-16
*** Standard errors may be too high due to more than 2 groups and exactDOF=FALSE
```

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 (2). 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 way 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   0.855295895
f1.2   0.323043910
f1.3  -0.146408669
f1.4  -1.304526972
f1.5  -1.210151012
f1.6  -0.852878418
f1.7  -0.646232831
f2.1   0.000000000
f2.2   0.002497546
f2.3  -0.602876990
f2.4   1.133586033
f2.5   0.346222171
f2.6  -0.043523593
f2.7   0.425860658
f2.8   0.445270489
f3.1   0.000000000
f3.2   0.068917817
f3.3   0.587689891
f3.4   0.295036592
```

```
f3.5  -0.052249653
f3.6   0.618678759
f3.7  -0.212497627
f3.8  -0.017318264
f3.9  -0.571389626
f3.10  0.782763900
```

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      0.53225199  16    1 f1     1
f1.2      0.00000000  17    1 f1     2
f1.3     -0.46945258  15    1 f1     3
f1.4     -1.62757088  12    1 f1     4
f1.5     -1.53319492  12    1 f1     5
f1.6     -1.17592233  15    1 f1     6
f1.7     -0.96927674  13    1 f1     7
f2.0.125  0.61808050  10    1 f2 0.125
f2.0.25   0.62057805  16    1 f2  0.25
f2.0.375  0.01520351  15    1 f2 0.375
f2.0.5    1.75166654  13    1 f2   0.5
f2.0.625  0.96430267  12    1 f2 0.625
f2.0.75   0.57455691  14    1 f2  0.75
f2.0.875  1.04394116  10    1 f2 0.875
f2.1      1.06335099  10    1 f2     1
f3.0.1   -0.29503660   5    2 f3   0.1
f3.0.2   -0.22611878   9    2 f3   0.2
f3.0.3    0.29265330  10    2 f3   0.3
f3.0.4    0.00000000  13    2 f3   0.4
f3.0.5   -0.34728625  11    2 f3   0.5
f3.0.6    0.32364216   8    2 f3   0.6
f3.0.7   -0.50753422   8    2 f3   0.7
f3.0.8   -0.31235486  13    2 f3   0.8
f3.0.9   -0.86642622  12    2 f3   0.9
f3.1      0.48772731  11    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)
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: 0x661e240>
<environment: 0x6d863d8>
```

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.

Here is the above example, but we create an intercept instead, and don't report the zero-coefficients, so that it closely resembles the output from `lm`

```
> f1 <- factor(f1); f2 <- factor(f2); f3 <- factor(f3)
> ef <- function(gamma,addnames) {
+   ref1 <- gamma[[1]]
+   ref2 <- gamma[[8]]
+   ref3 <- gamma[[16]]
+   # put the intercept in the first coordinate
+   gamma[[1]] <- ref1+ref2+ref3
+   gamma[2:7] <- gamma[2:7]-ref1
+   gamma[8:14] <- gamma[9:15]-ref2
+   gamma[15:23] <- gamma[17:25]-ref3
+   length(gamma) <- 23
+   if(addnames) {
+     names(gamma) <- c('(Intercept)',paste('f1',levels(f1)[2:7],sep=''),
+                       paste('f2',levels(f2)[2:8],sep=''),
+                       paste('f3',levels(f3)[2:10],sep=''))
+   }
+   gamma
+ }
> getfe(est,ef=ef,bN=1000,se=TRUE)
                  effect        se
(Intercept)  0.855295903 0.6580799
f12         -0.532251987 0.3754300
f13         -1.001704566 0.3997481
f14         -2.159822874 0.4704065
f15         -2.065446909 0.4173907
f16         -1.708174311 0.3850300
f17         -1.501528728 0.4339135
f20.25       0.002497546 0.4267723
f20.375     -0.602876990 0.4465194
f20.5        1.133586032 0.4565473
f20.625      0.346222166 0.4795252
f20.75      -0.043523593 0.4328772
f20.875      0.425860660 0.5289203
f21          0.445270485 0.4800787
```

```
f30.2        0.068917811 0.6143404
f30.3        0.587689886 0.6300515
f30.4        0.295036587 0.6098390
f30.5       -0.052249658 0.5947964
f30.6        0.618678755 0.6528433
f30.7       -0.212497630 0.6462748
f30.8       -0.017318268 0.5840338
f30.9       -0.571389632 0.5805968
f31          0.782763894 0.6231075
```

```
> #compare with lm
> summary(lm(y ~ x1 + f1 + f2 + f3))
```

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

Residuals:
     Min       1Q   Median       3Q      Max
-1.88686 -0.72519 -0.07878  0.75584  2.30499

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.855296   0.714043   1.198 0.234709
x1           2.354480   0.111959  21.030  < 2e-16 ***
f12         -0.532252   0.401350  -1.326 0.188760
f13         -1.001705   0.424620  -2.359 0.020891 *
f14         -2.159823   0.506491  -4.264 5.70e-05 ***
f15         -2.065447   0.458213  -4.508 2.34e-05 ***
f16         -1.708174   0.417543  -4.091 0.000106 ***
f17         -1.501529   0.452579  -3.318 0.001395 **
f20.25       0.002498   0.469734   0.005 0.995772
f20.375     -0.602877   0.483709  -1.246 0.216459
f20.5        1.133586   0.502050   2.258 0.026821 *
f20.625      0.346222   0.503912   0.687 0.494131
f20.75      -0.043524   0.480941  -0.090 0.928131
f20.875      0.425861   0.558136   0.763 0.447823
f21          0.445270   0.535713   0.831 0.408479
f30.2        0.068918   0.641433   0.107 0.914720
f30.3        0.587690   0.633119   0.928 0.356219
f30.4        0.295037   0.636733   0.463 0.644430
f30.5       -0.052250   0.624295  -0.084 0.933520
f30.6        0.618679   0.666068   0.929 0.355907
f30.7       -0.212498   0.643462  -0.330 0.742126
f30.8       -0.017318   0.600979  -0.029 0.977086
f30.9       -0.571390   0.617176  -0.926 0.357474
f31          0.782764   0.639550   1.224 0.224759
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.076 on 76 degrees of freedom
```

```
Multiple R-squared:  0.9005,        Adjusted R-squared:  0.8704
F-statistic: 29.91 on 23 and 76 DF,  p-value: < 2.2e-16
```

5. Non-estimability

We 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. Note that this is an illustration of non-obvious non-estimability which may occur in much larger datasets, the author does not endorse using this kind of model for the kind of data you find below.

```
> set.seed(128)
> x1 <- rnorm(25)
> f1 <- sample(9,length(x1),replace=TRUE)
> f2 <- sample(8,length(x1),replace=TRUE)
> f3 <- sample(8,length(x1),replace=TRUE)
> e1 <- sin(f1) + 0.02*f2^2  + 0.17*f3^3 + rnorm(length(x1))
> 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
-8.703e-01 -3.681e-01 -5.249e-13  1.484e-01  1.884e+00

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   1.4080     0.4698   2.997   0.0302 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.303 on 5 degrees of freedom
Multiple R-squared: 0.9997   Adjusted R-squared: 0.9986
F-statistic: 932.5 on 19 and 5 DF, p-value: 1.344e-07
*** Standard errors may be too high due to more than 2 groups and exactDOF=FALSE
```

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

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

```
[1] FALSE
```

Indeed, the rank-deficiency is larger than expected. 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 (2), and its (column) rank deficiency is larger than 2.

```
> f1 <- factor(f1); f2 <- factor(f2); f3 <- factor(f3)
> D <- t(do.call('rBind',
```

```
+         lapply(list(f1,f2,f3),as,Class='sparseMatrix')))
> dim(D)
[1] 25 21
> as.integer(rankMatrix(D))
[1] 18
```

Alternatively we can use an internal function in lfe for finding the rank deficiency directly.

```
> lfe::rankDefic(list(f1,f2,f3))
[1] 3
```

This rank-deficiency also has an impact on the standard errors computed by `felm`. If the rank-deficiency is small relative to the degrees of freedom the standard errors are scaled slightly upwards if we ignore the rank deficiency, but if it is large, the impact on the standard errors can be substantial. The above mentioned rank-computation procedure can be activated by specifying `exactDOF=TRUE` in the call to `felm`, but it may be time-consuming if the factors have many levels. Computing the rank does not in itself help us find estimable functions for `getfe`.

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

Residuals:
       Min         1Q     Median         3Q        Max
-8.703e-01 -3.681e-01 -5.249e-13  1.484e-01  1.884e+00

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   1.4080     0.4289   3.283   0.0168 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.189 on 6 degrees of freedom
Multiple R-squared: 0.9997   Adjusted R-squared: 0.9988
F-statistic:  1181 on 18 and 6 DF, p-value: 3.699e-09
```

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

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

Residuals:
       Min         1Q     Median         3Q        Max
-8.703e-01 -3.681e-01 -4.175e-16  1.484e-01  1.884e+00

Coefficients:
```

```
    Estimate Std. Error t value Pr(>|t|)
x1    1.4080     0.4289   3.283   0.0168 *
f32   0.6968     1.0579   0.659   0.5346
f33   4.2314     1.5438   2.741   0.0337 *
f34       NA         NA      NA       NA
f35  19.4888     1.1596  16.807 2.83e-06 ***
f36  35.6836     2.2637  15.764 4.13e-06 ***
f37  57.7721     1.1852  48.745 5.00e-09 ***
f38  86.4976     1.2263  70.535 5.46e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.189 on 6 degrees of freedom
Multiple R-squared: 0.9997   Adjusted R-squared: 0.9988
F-statistic:  1181 on 18 and 6 DF, p-value: 3.699e-09
```

```
> getfe(est)
```

```
          effect obs comp fe idx
f1.1 -34.5246100   4    1 f1   1
f1.2 -33.3981895   5    1 f1   2
f1.3 -34.6697810   6    1 f1   3
f1.5 -27.3871015   1    1 f1   5
f1.6 -35.7423130   5    1 f1   6
f1.7 -36.4467839   1    1 f1   7
f1.9 -32.2794852   3    1 f1   9
f2.2  -0.4036936   3    1 f2   2
f2.3   1.8848221   3    1 f2   3
f2.5  -2.6807139   2    1 f2   5
f2.6   0.0000000   8    1 f2   6
f2.7   0.8925205   4    1 f2   7
f2.8   0.9552711   5    1 f2   8
```

In this particular example, we may use a different order of the factors, and we see that by partitioning the dataset on the factors `f1,f3` instead of `f1,f2`, there are 2 connected components (the factor `f2` gets its own `comp`-code, but this is not a graph theoretic component number, it merely indicates that there is a separate reference among these).

```
> summary(est <- felm(y ~ x1 + G(f1) + G(f3) + G(f2), exactDOF=TRUE))
```

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

Residuals:
       Min         1Q     Median         3Q        Max
-8.703e-01 -3.681e-01 -8.633e-13  1.484e-01  1.884e+00

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   1.4080     0.4289   3.283   0.0168 *
---
```

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

Residual standard error: 1.189 on 6 degrees of freedom
Multiple R-squared: 0.9997   Adjusted R-squared: 0.9988
F-statistic:  1181 on 18 and 6 DF, p-value: 3.699e-09
> is.estimable(efactory(est),est$fe)
[1] TRUE
> getfe(est)
          effect obs comp fe idx
f1.1   0.1451709   4    1 f1   1
f1.2   1.2715915   5    1 f1   2
f1.3   0.0000000   6    1 f1   3
f1.5   0.0000000   1    2 f1   5
f1.6  -1.0725321   5    1 f1   6
f1.7  -1.7770027   1    1 f1   7
f1.9   2.3902960   3    1 f1   9
f3.1 -34.6697808   4    1 f3   1
f3.2 -33.9730301   4    1 f3   2
f3.3 -30.4384044   1    1 f3   3
f3.4 -27.3871016   1    2 f3   4
f3.5 -15.1809403   4    1 f3   5
f3.6   1.0137745   2    1 f3   6
f3.7  23.1023365   3    1 f3   7
f3.8  51.8278127   6    1 f3   8
f2.2  -0.4036936   3    3 f2   2
f2.3   1.8848219   3    3 f2   3
f2.5  -2.6807138   2    3 f2   5
f2.6   0.0000000   8    3 f2   6
f2.7   0.8925205   4    3 f2   7
f2.8   0.9552711   5    3 f2   8
>
```

Below is the same estimation in `lm`. We see that the coefficient for `x1` is identical to the one from `felm`, but there is no obvious relation between e.g. the coefficients for `f1`; the difference `f15-f16` is not the same for `lm` and `felm`. Since these are in different components, they are not comparable. But of course, if we compare in the same component, e.g. `f16-f17` or take a combination which actually occurs in the dataset, it is unique (estimable):

```
> data.frame(f1,f2,f3)[1,]
  f1 f2 f3
1  1  6  7
```

I.e. if we add the coefficients `f1.1 + f2.6 + f3.7` and include the intercept for `lm`, we will get the same number for both `lm` and `felm`.

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

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.8703 -0.3681  0.0000  0.1484  1.8842

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -34.9283     0.9814 -35.590 3.28e-08 ***
x1            1.4080     0.4289   3.283  0.01677 *
f12           1.1264     1.3539   0.832  0.43726
f13          -0.1452     1.1624  -0.125  0.90469
f15           7.1375     1.8223   3.917  0.00783 **
f16          -1.2177     1.4116  -0.863  0.42147
f17          -1.9222     2.2396  -0.858  0.42369
f19           2.2451     1.9249   1.166  0.28772
f23           2.2885     1.5620   1.465  0.19325
f25          -2.2770     1.4553  -1.565  0.16871
f26           0.4037     1.1870   0.340  0.74538
f27           1.2962     1.5285   0.848  0.42894
f28           1.3590     1.4770   0.920  0.39302
f32           0.6968     1.0579   0.659  0.53457
f33           4.2314     1.5438   2.741  0.03370 *
f34               NA         NA      NA       NA
f35          19.4888     1.1596  16.807 2.83e-06 ***
f36          35.6836     2.2637  15.764 4.13e-06 ***
f37          57.7721     1.1852  48.745 5.00e-09 ***
f38          86.4976     1.2263  70.535 5.46e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.189 on 6 degrees of freedom
Multiple R-squared:  0.9997,      Adjusted R-squared:  0.9989
F-statistic:  1181 on 18 and 6 DF,  p-value: 3.699e-09
```

6. Weeks-Williams partitions

There is a partial solution to the non-estimability problem in [**8**]. Their idea is to partition the dataset into components in which all differences between factor levels are estimable. The components are connected components of a subgraph of an e-dimensional grid graph where e is the number of factors. That is, a graph is constructed with the observations as vertices, two observations are adjacent (in a graph theoretic sense) if they differ in at most one of the factors. The dataset is then partitioned into (graph theoretic) connected components. It's a finer partitioning than the above, and consequently introduces more reference levels than is necessary for identification. I.e. it does not find all estimable functions, but in some cases (e.g. in [**7**]) the largest component will be sufficiently large for proper analysis. It is of course always a question whether such an endogeneous selection of observations will yield a dataset which results in unbiased coefficients. This partitioning can be done by the `compfactor` function with argument `WW=TRUE`:

```
> fe <- list(f1,f2,f3)
> wwcomp <- compfactor(fe, WW=TRUE)
```

It has more levels than the rank deficiency

```
> lfe::rankDefic(fe)
[1] 3
> nlevels(wwcomp)
[1] 10
```

and each of its components are contained in a component of the previously considered components, no matter which two factors we consider. For the case of two factors, the concepts coincide.

```
> nlevels(interaction(compfactor(fe),wwcomp))
[1] 10
> # pick the largest component:
> wwdata <- data.frame(y, x1, f1, f2, f3)[wwcomp==1, ]
> print(wwdata)
           y           x1 f1 f2 f3
2   52.90863  0.482611999  2  6  8
8   54.99005  0.004552681  2  6  8
11 -13.28338  0.322709391  2  8  5
15  54.61962  0.684795317  2  7  8
16 -35.09158 -0.883670355  3  6  2
17  56.12736  0.722126561  9  7  8
19 -14.59621 -0.563255192  2  6  5
20 -18.93693 -1.644454913  6  6  5
21 -34.36695  0.359965899  3  2  2
25  52.12814  0.589059932  3  6  8
```

Though, in this particular example, there are more parameters than there are observations, so an estimation would not be feasible.

`efactory` cannot easily be modified to produce an estimable function corresponding to WW components. The reason is that `efactory`, and the logic in `getfe`, work on partitions of factor levels, not on partitions of the dataset, these are the same for the two-factor case.

WW partitions have the property that if you pick any two of the factors and partition a WW-component into the previously mentioned non-WW partitions, there will be only one component, hence you may use any of the estimable functions from `efactory` on each partition. That is, a way to use WW partitions with **lfe** is to do the whole analysis on the largest WW-component. `felm` may still be used on the whole dataset, and it may yield different results than what you get by analysing the largest WW-component.

Here is a larger example:

```
> set.seed(135)
> x <- rnorm(10000)
> f1 <- sample(1000,length(x),replace=TRUE)
> f2 <- (f1 + sample(18,length(x), replace=TRUE)) %% 500
> f3 <- (f2 + sample(9,length(x),replace=TRUE)) %% 500
> y <- x + 1e-4*f1 + sin(f2^2) +
```

```
+   cos(f3)^3 + 0.5*rnorm(length(x))
> dataset <- data.frame(y,x,f1,f2,f3)
> summary(est <- felm(y ~ x + G(f1) + G(f2) + G(f3),
+              data=dataset, exactDOF=TRUE))
Call:
   felm(formula = y ~ x + G(f1) + G(f2) + G(f3), data = dataset,      exactDOF = TRUE)

Residuals:
      Min        1Q    Median        3Q       Max
-1.929087 -0.305385  0.000243  0.305398  1.921153

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x 0.999908   0.005646   177.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5053 on 8001 degrees of freedom
Multiple R-squared: 0.9035   Adjusted R-squared: 0.8794
F-statistic: 37.48 on 1998 and 8001 DF, p-value: < 2.2e-16
```

We count the number of connected components in `f1,f2`, and see that this is sufficient to ensure estimability

```
> nlevels(est$cfactor)
[1] 1
> is.estimable(efactory(est), est$fe)
[1] TRUE
> nrow(alpha <- getfe(est))
[1] 2000
```

It has rank deficiency one less than the number of factors :

```
> lfe:::rankDefic(est$fe)
[1] 2
```

Then we analyse the largest WW-component

```
> wwcomp <- compfactor(est$fe,WW=TRUE)
> nlevels(wwcomp)
[1] 958
> wwset <- dataset[wwcomp == 1, ]
> nrow(wwset)
[1] 2394
> summary(wwest <- felm(y ~ x + G(f1) + G(f2) + G(f3),
+              data=wwset, exactDOF=TRUE))
Call:
   felm(formula = y ~ x + G(f1) + G(f2) + G(f3), data = wwset, exactDOF = TRUE)

Residuals:
       Min         1Q     Median         3Q        Max
```

```
-1.8828643 -0.2949762  0.0001938  0.2943913  2.0374621

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x  1.00290    0.01201   83.51   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5138 on 1788 degrees of freedom
Multiple R-squared: 0.9111   Adjusted R-squared: 0.8809
F-statistic: 30.28 on 605 and 1788 DF, p-value: < 2.2e-16
```

We see that we get the same coefficient for `x` in this case. This is not surprising, there is no obvious reason to believe that our selection of observations is skewed in this randomly created dataset.

This one has the same rank deficiency:

```
> lfe::rankDefic(wwest$fe)
[1] 2
```

but a smaller number of identifiable coefficients.

```
> nrow(wwalpha <- getfe(wwest))
[1] 607
```

We may compare effects which are common to the two methods:

```
> head(alpha)
         effect obs comp fe idx
f1.1 -0.5541796   6    1 f1   1
f1.2 -0.2755604  13    1 f1   2
f1.3 -0.5763353   6    1 f1   3
f1.4 -0.5413368   8    1 f1   4
f1.5 -0.5696979  13    1 f1   5
f1.6 -0.4250255   9    1 f1   6
> head(wwalpha)
          effect obs comp fe idx
f1.1 -1.27887479   3    1 f1   1
f1.2 -0.34865371   7    1 f1   2
f1.3 -0.95120588   4    1 f1   3
f1.4 -1.04718930   1    1 f1   4
f1.6 -0.89773364   4    1 f1   6
f1.7  0.04965163   1    1 f1   7
```

but there is no obvious relation between e.g. `f1.1 - f1.2`, they are very different in the two estimations. The coefficients are from different datasets, and the standard errors are large (≈ 0.7) with this few observations for each factor level. The number of identified coefficients for each factor varies (these figures contain the two references):

```
> table(wwalpha[,'fe'])
 f1  f2  f3
310 148 149
```

References

[1] J. M. Abowd, R. H. Creecy, and F. Kramarz, *Computing Person and Firm Effects Using Linked Longitudinal Employer-Employee Data*, Tech. Report TP-2002-06, U.S. Census Bureau, 2002.

[2] J. M. Abowd, F. Kramarz, and D. N. Margolis, *High Wage Workers and High Wage Firms*, Econometrica **67** (1999), no. 2, 251–333.

[3] J. A. Eccleston and A. Hedayat, *On the Theory of Connected Designs: Characterization and Optimality*, Ann. Statist. **2** (1974), 1238–1255.

[4] S. Gaure, *OLS with Multiple High Dimensional Category Variables*, Computational Statistics and Data Analysis **66** (2013), 8–18.

[5] J. D. Godolphin and E. J. Godolphin, *On the connectivity of row-column designs*, Util. Math. **60** (2001), 51–65.

[6] L.L. Kupper, J.M. Janis, I.A. Salama, C.N. Yoshizawa, and B.G. Greenberg, *Age-Period-Cohort Analysis: An Illustration of the Problems in Assessing Interaction in One Observation Per Cell Data*, Commun. Statist.-Theor. Meth. **12** (1983), no. 23, 2779–2807.

[7] S.M. Torres, P. Portugal, J.T. Addison, and P. Guimarães, *The Sources of Wage Variation: A Three-Way High-Dimensional Fixed Effects Regression Model.*, IZA Discussion Paper 7276, Institute for the Study of Labor (IZA), March 2013.

[8] D.L. Weeks and D.R. Williams, *A Note on the Determination of Connectedness in an N-Way Cross Classification*, Technometrics **6** (1964), no. 3, 319–324.

Ragnar Frisch Centre for Economic Research, Oslo, Norway