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  0.37627519   2    1 f1 0.1
f1.0.2 -0.08109998   1    2 f1 0.2
f1.0.3 -0.68688030   3    1 f1 0.3
f1.0.4  0.57317750   4    1 f1 0.4
f1.0.5  0.47914188   2    1 f1 0.5
f1.0.6  1.41301954   3    1 f1 0.6
```

```
f1.0.7  0.84495593   1    2 f1 0.7
f1.0.8  0.92643381   4    1 f1 0.8
f2.0.1 -0.00401133   3    1 f2 0.1
f2.0.2  0.00000000   5    1 f2 0.2
f2.0.3 -1.51866658   1    1 f2 0.3
f2.0.4  0.00000000   2    2 f2 0.4
f2.0.5 -1.89452369   2    1 f2 0.5
f2.0.6 -0.88431922   3    1 f2 0.6
f2.0.7 -0.60911027   3    1 f2 0.7
f2.0.8 -0.96865247   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 $\hat{\beta}$. 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 (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
-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 (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   0.855295903
f1.2   0.323043918
f1.3  -0.146408669
f1.4  -1.304526974
f1.5  -1.210151022
f1.6  -0.852878427
f1.7  -0.646232814
f2.1   0.000000000
f2.2   0.002497552
f2.3  -0.602876984
f2.4   1.133586021
f2.5   0.346222168
f2.6  -0.043523600
f2.7   0.425860665
f2.8   0.445270478
f3.1   0.000000000
f3.2   0.068917820
f3.3   0.587689884
f3.4   0.295036588
```

```
f3.5  -0.052249655
f3.6   0.618678760
f3.7  -0.212497631
f3.8  -0.017318264
f3.9  -0.571389617
f3.10  0.782763895
```

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     -0.46945259  15    1 f1     3
f1.4     -1.62757089  12    1 f1     4
f1.5     -1.53319494  12    1 f1     5
f1.6     -1.17592234  15    1 f1     6
f1.7     -0.96927673  13    1 f1     7
f2.0.125  0.61808051  10    1 f2 0.125
f2.0.25   0.62057806  16    1 f2  0.25
f2.0.375  0.01520352  15    1 f2 0.375
f2.0.5    1.75166653  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  1.04394117  10    1 f2 0.875
f2.1      1.06335098  10    1 f2     1
f3.0.1   -0.29503659   5    2 f3   0.1
f3.0.2   -0.22611877   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   -0.34728624  11    2 f3   0.5
f3.0.6    0.32364217   8    2 f3   0.6
f3.0.7   -0.50753422   8    2 f3   0.7
f3.0.8   -0.31235485  13    2 f3   0.8
f3.0.9   -0.86642621  12    2 f3   0.9
f3.1      0.48772730  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,'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: 0x5bc9120>
<environment: 0x629ec60>
```

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

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.

```
> 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 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,'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(do.call('rBind',
+         lapply(list(f1,f2,f3),as,Class='sparseMatrix')))
> dim(D)
[1] 100  99
> as.integer(rankMatrix(D))
[1] 92
> # alternatively we can use an internal function
> # in lfe for finding the rank deficiency directly
> lfe::rankDefic(list(f1,f2,f3))
[1] 7
```

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, as in this example, the effect on the standard errors may be substantial. The 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
-8.690e-01 -9.853e-02 -9.920e-12  1.135e-01  8.690e-01

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x1   1.6543     0.4795    3.45   0.0107 *
---
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
```

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
```

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 `f12-f13` is not the same for `lm` and `felm`. But of course, if we take a combination which actually occurs in the dataset, it is estimable:

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

```
  f1    f2  f3
1 31 2.125 0.1
```

I.e. if we add the coefficients `f1.31 + f2.2.125 + f3.0.1` 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
-0.86895 -0.09853  0.00000  0.11346  0.86895

Coefficients: (5 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.66103    3.20506  -1.142  0.29091
x1           1.65426    0.47951   3.450  0.01069 *
f12          5.55766    2.26821   2.450  0.04409 *
f13         -0.50215    1.82413  -0.275  0.79105
f14         -6.22264    3.01472  -2.064  0.07789 .
f15         -3.25066    1.30713  -2.487  0.04179 *
f16         -0.90207    1.43495  -0.629  0.54954
f17         -1.94779    2.31183  -0.843  0.42734
f18          1.06828    2.19941   0.486  0.64200
```

f19	-3.71630	1.74689	-2.127	0.07094	.
f110	7.46271	5.20570	1.434	0.19481	
f111	-2.79296	2.03317	-1.374	0.21191	
f112	-2.39955	1.22205	-1.964	0.09035	.
f113	3.04482	2.64864	1.150	0.28807	
f114	2.26528	1.84794	1.226	0.25989	
f115	0.50911	2.17930	0.234	0.82197	
f116	0.77581	1.84701	0.420	0.68705	
f117	-1.73116	1.45181	-1.192	0.27195	
f118	-2.38905	2.70538	-0.883	0.40650	
f119	-0.10752	1.42174	-0.076	0.94183	
f120	-1.78120	1.96692	-0.906	0.39526	
f121	2.40789	1.95402	1.232	0.25763	
f122	0.57434	2.66231	0.216	0.83535	
f123	5.34066	2.63774	2.025	0.08255	.
f125	-3.10254	2.41876	-1.283	0.24043	
f126	5.77565	1.87303	3.084	0.01773	*
f127	-0.98631	2.89668	-0.340	0.74347	
f128	-0.54472	1.98226	-0.275	0.79140	
f129	1.10020	2.85622	0.385	0.71153	
f130	-4.42386	2.01494	-2.196	0.06415	.
f131	-0.31554	1.40158	-0.225	0.82831	
f132	1.67510	1.87694	0.892	0.40178	
f133	-0.04469	1.58114	-0.028	0.97824	
f134	0.23692	2.21817	0.107	0.91794	
f20.25	-0.21878	3.33463	-0.066	0.94952	
f20.375	-2.18662	3.85525	-0.567	0.58831	
f20.5	2.36500	3.09068	0.765	0.46916	
f20.625	-2.23073	3.71307	-0.601	0.56692	
f20.75	0.72413	2.40222	0.301	0.77184	
f20.875	7.27337	3.08912	2.355	0.05075	.
f21	-0.48939	2.85158	-0.172	0.86859	
f21.25	0.64844	2.86718	0.226	0.82754	
f21.375	4.18909	2.63493	1.590	0.15590	
f21.5	0.72470	2.57333	0.282	0.78638	
f21.625	0.87861	2.48292	0.354	0.73386	
f21.75	-0.60505	2.54223	-0.238	0.81870	
f21.875	3.61218	2.83350	1.275	0.24306	
f22	-1.32018	3.41091	-0.387	0.71022	
f22.125	2.02921	2.68218	0.757	0.47401	
f22.25	3.51098	3.11126	1.128	0.29631	
f22.375	3.43161	3.17004	1.083	0.31490	
f22.5	3.56781	2.20129	1.621	0.14910	
f22.625	4.50405	3.17600	1.418	0.19909	
f22.75	4.61179	4.49292	1.026	0.33883	
f22.875	0.71555	2.67108	0.268	0.79651	
f23	7.58317	3.97162	1.909	0.09785	.
f23.125	1.76329	3.16922	0.556	0.59528	

```
f23.25       2.28357    2.30731   0.990  0.35528
f23.5       -0.31742    2.54077  -0.125  0.90409
f23.625      1.91023    2.61134   0.732  0.48823
f23.75       2.10793    2.76506   0.762  0.47076
f23.875     -3.12685    3.27222  -0.956  0.37112
f24          0.89211    2.42757   0.367  0.72411
f24.125      0.61089    2.64833   0.231  0.82417
f24.25       0.53340    2.54879   0.209  0.84019
f30.2        0.67485    5.04441   0.134  0.89734
f30.3        3.27133    1.61575   2.025  0.08256 .
f30.4       -0.74433    2.25029  -0.331  0.75050
f30.5        1.71125    1.48109   1.155  0.28584
f30.6        1.39896    1.64910   0.848  0.42432
f30.7        5.22827    2.67470   1.955  0.09153 .
f30.8       -0.18433    2.13262  -0.086  0.93354
f30.9        1.33904    1.40802   0.951  0.37328
f31          2.28896    2.44370   0.937  0.38011
f31.1        1.26117    1.22790   1.027  0.33855
f31.2        0.78355    1.91729   0.409  0.69499
f31.3        3.50358    2.53627   1.381  0.20964
f31.4        1.70787    2.09532   0.815  0.44187
f31.5        1.65115    1.62800   1.014  0.34424
f31.6        4.06636    1.53549   2.648  0.03303 *
f31.7        2.75901    1.88728   1.462  0.18716
f31.8        5.69671    2.45122   2.324  0.05308 .
f31.9        2.81805    2.36233   1.193  0.27176
f32               NA         NA      NA       NA
f32.1             NA         NA      NA       NA
f32.2        3.43994    3.20692   1.073  0.31900
f32.3        5.09893    1.84553   2.763  0.02798 *
f32.4             NA         NA      NA       NA
f32.5        0.94775    2.34827   0.404  0.69855
f32.6             NA         NA      NA       NA
f32.7             NA         NA      NA       NA
f32.8        7.71093    2.09613   3.679  0.00787 **
f32.9        7.26614    1.92547   3.774  0.00695 **
f33          7.37830    1.81427   4.067  0.00477 **
f33.1        4.26468    1.53926   2.771  0.02767 *
f33.2        8.00990    1.96640   4.073  0.00473 **
f33.3        7.32992    2.01924   3.630  0.00840 **
f33.4       11.28521    2.35848   4.785  0.00200 **
---
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:  0.9408
F-statistic: 18.09 on 92 and 7 DF,  p-value: 0.0002557
```

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