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 + ... + xm + 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 + ... + xm | f1 + f2 + ... + 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 refer-
ence. 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 \texttt{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. \texttt{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 rela-
tionships, 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, identification is well-known. \texttt{getf} will partition
the dataset into connected components, and introduce a reference level in each
component:

```r
library(lfe)
#
## Call:
## felm(formula = y ~ x1 | f1 + f2)
##
## Residuals:
##    Min  1Q Median  3Q Max
## -0.7331 -0.1751  0.0000  0.1139  0.7331
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## x1            1.9609     0.2854   6.872  0.00097 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8097 on 5 degrees of freedom
## Multiple R-squared(full model): 0.9849  Adjusted R-squared: 0.9425
## Multiple R-squared(proj model): 0.9042  Adjusted R-squared: 0.6361
## F-statistic(full model):23.23 on 14 and 5 DF, p-value: 0.001318
## F-statistic(proj model): 47.22 on 1 and 5 DF, p-value: 0.0009982
```
We examine the estimable function produced by `efactory`.

```r
ef <- efactory(est)
is.estimable(ef, est$fe)
## [1] TRUE
getfe(est)
```

As we can see from the `comp` entry, there are two components, the second one with \( f1=0.1, f1=0.3, f1=0.5 \), and \( f2=0.5 \) and \( f2=0.7 \). A reference is introduced in each of the components, i.e. \( f2.0.2=0 \) and \( f2.0.5=0 \). If we look at the dataset, the component structure becomes clearer:

```r
data.frame(f1, f2, comp = est$cfactor)
```

---

**MULTICOLLINEARITY, IDENTIFICATION, AND ESTIMABLE FUNCTIONS**

3
## 19 0.7 0.1 1
## 20 0.1 0.7 2

Observations 1, 10, 12, 17, and 20 belong to component 2; no other observation has \( f_1 \) \( \in \{0.1, 0.3, 0.5\} \) or \( f_2 \) \( \in \{0.5, 0.7\} \), thus it is clear that coefficients for these can not be compared to other coefficients. \texttt{lm} is silent about this component structure, hence coefficients are hard to interpret. Though, predictive properties and residuals are the same:

```r
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
## 2.095e-01 7.331e-01 6.939e-17 -4.393e-01 -6.495e-01 -5.678e-01 -2.082e-17
## 8 9 10 11 12 13 14
## 1.388e-17 6.029e-01 -2.706e-16 8.197e-02 3.400e-16 4.859e-01 -1.636e-01
## 15 16 17 18 19 20
##
## **Coefficients:** (1 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.6742    0.8930   0.755  0.484267
## x1                         1.9609    0.2854  6.872  0.000998 ***
## f10.2                      -2.4282    1.4826  -1.638  0.162390
## f10.3                      -0.2382    1.5798  -0.151  0.886042
## f10.4                      -1.9502    1.6137  -1.208  0.280892
## f10.5                      -0.1680    1.5778  -0.105  0.915034
## f10.6                      -1.7645    1.6137  -1.053  0.307381
## f10.7                      -4.0375    1.5625  -2.504  0.049189 *
## f10.8                      -2.4642    1.5037  -1.639  0.212781
## f20.2                      2.1776    1.0249   2.125  0.086978 .
## f20.3                      2.6178    1.4837   1.764  0.137940
## f20.4                      1.2401    1.6508   0.751  0.486366
## f20.5                      0.1681    1.3269  0.127  0.904134
## f20.6                      1.5816    1.1080   1.428  0.212781
## f20.7                      NA         NA      NA    NA
## f20.8                      2.1528    1.3808   1.559  0.179716
##
## **Signif. codes:**  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 .’ 0.1 ’ ’ 1
##
## Residual standard error: 0.8097 on 5 degrees of freedom
## Multiple R-squared: 0.9849, Adjusted R-squared: 0.9425
## F-statistic: 23.23 on 14 and 5 DF, p-value: 0.001318
```
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. \texttt{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:

\begin{verbatim}
est <- felm(logwage ~ x1 + x2 | id + firm + edu) getfe(est)
\end{verbatim}

This will result in the same number of references as if using the model

\begin{verbatim}
logwage ~ x1 + x2 + edu | id + firm
\end{verbatim}

though it may run faster (or slower).
Alternatively, one could specify the model as

\begin{verbatim}
logwage ~ x1 + x2 | firm + edu + id
\end{verbatim}

This would not account for a partitioning 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 \texttt{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 \texttt{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 \texttt{lfe} in this respect. You will meet the same problem with \texttt{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. \texttt{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. \cite{6}) you will find that the coefficients are only identified up to linear trends. You may force the linear trend between \texttt{birth=1963} and \texttt{birth=1990} to zero, by removing the reference level \texttt{birth=1990}, and the system will be free of multicollinearities. In this case the \texttt{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 \texttt{age} and \texttt{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 \texttt{lm} or \texttt{getfe} reports a handful of non-obvious spurious multicollinearities 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

\[ y \sim x_1 + x_2 + f_1 + f_2 + f_3 \]

may be written in matrix notation as

\[ y = X\beta + D\alpha + \epsilon, \]

where \( X \) is a matrix with columns \( x_1 \) and \( x_2 \) and \( D \) is matrix of dummies constructed from the levels of the factors \( f_1, f_2, f_3 \). 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]):

\[ \text{est} \leftarrow \text{felm}(y \sim x_1 + x_2 | f_1 + f_2 + f_3) \]

This yields the parameters for \( x_1 \) and \( x_2 \), i.e. \( \hat{\beta} \). To find \( \hat{\alpha} \), the parameters for the levels of \( f_1, f_2, f_3 \), **getfe** solves a certain linear system (see [4, eq. (14)]):

\[ D\gamma = \rho \]

where the vector \( \rho \) can be computed when we have \( \hat{\beta} \). This does not identify \( \gamma \) uniquely, we have to apply an estimable function to \( \gamma \). The estimable function \( F \) is characterized by the property that \( F\gamma_1 = F\gamma_2 \) whenever \( \gamma_1 \) and \( \gamma_2 \) 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:
summary(est <- felm(y ~ x1 | f1 + f2 + f3))
##
## Call:
## felm(formula = y ~ x1 | f1 + f2 + f3)
##
## Residuals:
##    Min 1Q Median 3Q Max
## -2.18822 -0.55222 0.09278 0.62858 2.31181
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## x1 2.5538 0.1026 24.88 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.9963 on 76 degrees of freedom
## Multiple R-squared(full model): 0.9086 Adjusted R-squared: 0.8809
## Multiple R-squared(proj model): 0.8907 Adjusted R-squared: 0.8576
## F-statistic(full model):32.84 on 23 and 76 DF, p-value: < 2.2e-16
## F-statistic(proj model): 619.2 on 1 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 $\gamma$ 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 $f_2,f_3$, subtract these, and add the sum to the first factor:

```
ef <- function(gamma, addnames) {
  ref2 <- gamma[8]
  ref3 <- gamma[16]
  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

gtextfe(est, ef = ef)
##  effect
## f1.1 -0.013634682
## f1.2  0.727611420
## f1.3 -0.521386749
## f1.4 -0.46496809
## f1.5  1.568204155
## f1.6 -0.1511048
## f1.7  0.286980841
## f2.1  0.000000000
## f2.2 -0.289569658
## f2.3  0.168627982
## f2.4 -0.658310494
## f2.5  0.253613291
## f2.6  0.427094488
## f2.7 -0.249330433
## f2.8 -0.772323808
## f3.1  0.000000000
## f3.2 -0.004888500
## f3.3  0.168627982
## f3.4  0.449689498
## f3.5  0.729926376
## f3.6  0.697845803
## f3.7  0.569065140
## f3.8  0.583417051
## f3.9  0.113820998
## f3.10 0.005328265

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

gtextfe(est)
##  effect obs comp fe idx
## f1.1 -0.74124610 16  1 f1  1
## f1.2  0.00000000 19  1 f1  2
## f1.3 -1.24899817 15  1 f1  3
## f1.4 -1.37410823 12  1 f1  4
## f1.5 -2.29581558 10  1 f1  5
## f1.6 -0.87912247 16  1 f1  6
## f1.7 -0.44063058 12  1 f1  7
## f2.0.125 1.29667656 11  1 f2 0.125
## f2.0.25  1.00710691 15  1 f2 0.25
## f2.0.375  1.46530454 14  1 f2 0.375
## f2.0.5  0.63836607 11  1 f2 0.5
## f2.0.625 1.55028985 12  1 f2 0.625
## f2.0.75  1.72377105 12  1 f2 0.75
## f2.0.875 1.04734613 14  1 f2 0.875
We see that the default has some more information. It uses the level names, and some more information, added like this:

```r
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: 0x5a5c28f250f0>
## <environment: 0x5a5c28bcb4c8>
```

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`

```r
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
```
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) -0.013634682 0.5114435
## f12  0.741246102 0.3160189
## f13  -0.507752065 0.3373377
## f14  -0.632862126 0.3572938
## f15  -1.554569472 0.3811126
## f16  -0.137876368 0.3402928
## f17  0.300615524 0.3485067
## f20.25 -0.289569657 0.4047196
## f20.375  0.168627982 0.3873691
## f20.5  -0.658310496 0.463872
## f20.625  0.253613289 0.4170575
## f20.75  0.427094489 0.4318387
## f20.875  0.249330434 0.4006819
## f21  -0.772323808 0.4004357
## f30.2  -0.004888501 0.4376401
## f30.3  -0.205494035 0.4464411
## f30.4  0.449689499 0.4995956
## f30.5  0.729926375 0.4339349
## f30.6  0.697845804 0.4643069
## f30.7  0.569065141 0.4218616
## f30.8  0.583417050 0.4345362
## f30.9  0.113820992 0.5315648
## f31  0.005328265 0.4159621
#
# compare with lm
summary(lm(y ~ x1 + f1 + f2 + f3))
## Call:
## lm(formula = y ~ x1 + f1 + f2 + f3)
#### Residuals:
##    Min     1Q Median     3Q    Max
## 2.18822 0.55222 0.09278 0.62858 2.31181
## Coefficients:
### Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.013635 0.579452 -0.024 0.981289
x1 2.553781 0.102627 24.884 < 2e-16 ***
f12 0.741246 0.385522 1.923 0.058264 .
f13 -0.507752 0.393773 -1.289 0.201151
f14 -0.632862 0.420832 -1.504 0.136768
f15 -1.554569 0.444675 -3.496 0.000792 ***
f16 -0.137876 0.387709 -0.356 0.723111
f17 0.300616 0.417071 0.721 0.473257
f20.25 -0.289570 0.455406 -0.636 0.526785
f20.375 0.168628 0.457298 0.369 0.713340
f20.5 -0.658310 0.516023 -1.276 0.205934
f20.625 0.253613 0.475630 0.533 0.595090
f20.75 0.427094 0.503638 0.848 0.399090
f20.875 -0.249330 0.458179 -0.544 0.587913
f21 -0.772324 0.460361 -1.678 0.097524 .
f30.2 -0.004889 0.491509 -0.010 0.992091
f30.3 -0.205494 0.510660 -0.402 0.688513
f30.4 0.449689 0.567468 0.792 0.430565
f30.5 0.729926 0.504571 1.447 0.152113
f30.6 0.697846 0.546266 1.277 0.205320
f30.7 0.569065 0.466833 1.219 0.226667
f30.8 0.583417 0.473972 1.231 0.221521
f30.9 0.113821 0.584693 0.195 0.846172
f31 0.005328 0.467270 0.011 0.990932
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9963 on 76 degrees of freedom
Multiple R-squared: 0.9086, Adjusted R-squared: 0.8809
F-statistic: 32.84 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 $x_1$ 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.

```r
code here```

```r
set.seed(55)
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))
```
$$y \leftarrow 2.5 \times x_1 + (e_1 - \text{mean}(e_1))$$

```r
summary(est <- felm(y ~ x1 | f1 + f2 + f3))
```

```
##
## Call:
## felm(formula = y ~ x1 | f1 + f2 + f3)
##
## Residuals:
## Min 1Q Median 3Q Max
## -0.43725 -0.09946 0.00000 0.05047 0.38973
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## x1 0.9735 1.3111 0.743 0.593
##
## Residual standard error: 1.146 on 1 degrees of freedom
## Multiple R-squared(full model): 0.9999 Adjusted R-squared: 0.9977
## Multiple R-squared(proj model): 0.3554 Adjusted R-squared: -14.47
## F-statistic(full model): 447.3 on 23 and 1 DF, p-value: 0.03731
## F-statistic(proj model): 0.5514 on 1 and 1 DF, p-value: 0.5934
## *** 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 usable. `getfe` yields a warning in this case.

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

```
## Warning in is.estimable(ef, est$fe): non-estimable function, largest error 2e-04 in coordinate 4 ("f1.4")
## [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.

```r
f1 <- factor(f1)
f2 <- factor(f2)
f3 <- factor(f3)
D <- makeDmatrix(list(f1, f2, f3))
dim(D)
```

```
## [1] 25 25
```

```r
ncol(D) - as.integer(rankMatrix(D))
```

```
## [1] 3
```

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

```r
lfe:::rankDefic(list(f1, f2, f3))
```

```
## [1] 3
```
This rank-deficiency also has an impact on the standard errors computed by \texttt{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 \texttt{exactDOF=TRUE} in the call to \texttt{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 \texttt{getfe}.

\begin{verbatim}
summary(est <- felm(y ~ x1 | f1 + f2 + f3, exactDOF = TRUE))
## Call:
##   felm(formula = y ~ x1 | f1 + f2 + f3, exactDOF = TRUE)
## ## Residuals:
##     Min  1Q Median  3Q Max
## -0.43725 -0.09946  0.00000  0.05047  0.38973
## ## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x1   0.9735     0.9271  1.05   0.403
#### Residual standard error: 0.8105 on 2 degrees of freedom
## Multiple R-squared(full model): 0.9999 Adjusted R-squared: 0.9988
## Multiple R-squared(proj model): 0.3554 Adjusted R-squared: -6.735
## F-statistic(full model):935.2 on 22 and 2 DF, p-value: 0.001069
## F-statistic(proj model): 1.103 on 1 and 2 DF, p-value: 0.4038
\end{verbatim}

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

\begin{verbatim}
summary(est <- felm(y ~ x1 + f3 | f1 + f2, exactDOF = TRUE))
## Warning in chol.default(mat, pivot = TRUE, tol = tol): the matrix is either rank-deficient or not positive definite
## Warning in chol.default(mat, pivot = TRUE, tol = tol): the matrix is either rank-deficient or not positive definite
## Call:
##   felm(formula = y ~ x1 + f3 | f1 + f2, exactDOF = TRUE)
## ## Residuals:
##     Min  1Q Median  3Q Max
## -0.43725 -0.09946  0.00000  0.05047  0.38973
## ## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x1   0.9735     0.9271  1.050  0.404
\end{verbatim}
In this particular example, we may use a different order of the factors, and we see that by partitioning the dataset on the factors \( f_1, f_3 \) instead of \( f_1, f_2 \), there are 2 connected components (the factor \( f_2 \) 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).

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

```r
## Call:
## felm(formula = y ~ x1 | f1 + f3 + f2, exactDOF = TRUE)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##    NaN     NaN     NaN     NaN     NaN
```
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 f14-f15 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 2 6 3

I.e. if we add the coefficients \( f_{1.2} + f_{2.6} + f_{3.3} \) and include the intercept for \( \text{lm} \), we will get the same number for both \( \text{lm} \) and \( \text{felm} \). That is, for predicting the actual dataset, estimability plays no role, we obtain the same residuals anyway. It is only for predicting outside of the dataset estimability is important.

summary(est <- lm(y ~ x1 + f1 + f2 + f3))
## Call:
## lm(formula = y ~ x1 + f1 + f2 + f3)
## ## Residuals:
## 1 2 3 4 5 6 7
## 3.883e-01 -2.873e-01 -4.899e-02 1.485e-01 3.378e-01 -1.388e-17 -5.047e-02
## 8 9 10 11 12 13 14
## 15 16 17 18 19 20 21
## 1.388e-17 5.551e-17 4.899e-02 -1.485e-01 -2.776e-17 3.897e-01 -4.899e-02
## 22 23 24 25
## -2.398e-01 -3.393e-01 -9.714e-17 -9.946e-02
## # Coefficients: (1 not defined because of singularities)
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.3886 1.1202 -21.772 0.002103 **
## x1 0.9735 0.9271 1.050 0.403842
## f12 -1.5133 1.2712 -1.190 0.356003
## f13 -0.4383 1.0229 -0.429 0.710016
## f14 79.1550 2.2569 35.073 0.000812 ***
## f15 -3.4261 1.2614 -2.716 0.113027
## f16 1.3450 2.8879 0.466 0.687194
## f17 -0.2386 0.9916 -0.241 0.832255
## f18 -0.5701 2.0710 -0.275 0.808947
## f19 -2.2171 1.1201 -1.979 0.186330
## f22 -0.2337 2.2869 -0.102 0.927917
## f23 3.9358 2.7071 1.454 0.283177
## f24 -3.8157 3.1342 -1.217 0.347584
## f25 -1.0591 1.1201 -0.946 0.352509
## f26 0.2702 0.9701 0.278 0.806791
## f27 1.3360 1.1119 1.202 0.352520
## f28 3.6488 1.4917 2.446 0.134276
## f32 0.4317 1.0394 0.415 0.718239
## f33 5.1696 1.1358 4.552 0.045034 *
## f34 9.8295 2.1756 4.518 0.045659 *
## f35 19.0791 1.3493 14.140 0.004964 **
## f36 34.7134 2.3414 14.826 0.004519 **
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 endogenous 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`:

```r
fe <- list(f1, f2, f3)
wcomp <- compfactor(fe, WW = TRUE)
```

It has more levels than the rank deficiency

```r
lfe:::rankDefic(fe)
## [1] 3
nlevels(wcomp)
## [1] 17
```

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.

```r
nlevels(interaction(compfactor(fe), wcomp))
## [1] 17
```

# pick the largest component:
```r
wwdata <- data.frame(y, x1, f1, f2, f3)[wcomp == 1, ]
print(wwdata)
```
That is, we can start in one of the observations and travel through all of them by changing just one of \(f_1, f_2, f_3\) at a time. 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:

```r
define 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 | f1 + f2 + f3, data = dataset, exactDOF = TRUE))
```

```
Call:
felm(formula = y ~ x | f1 + f2 + f3, data = dataset, exactDOF = TRUE)

Residuals:
  Min      1Q  Median      3Q     Max
-1.63055 -0.29857 -0.00236  0.30599  1.79423

Coefficients:
  Estimate Std. Error t value Pr(>|t|)    
x  0.998552  0.005548 180   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4957 on 8001 degrees of freedom
Multiple R-squared(full model): 0.9058 Adjusted R-squared: 0.8822
Multiple R-squared(proj model): 0.8019 Adjusted R-squared: 0.7524
F-statistic(full model):38.49 on 1998 and 8001 DF, p-value: < 2.2e-16
```
We count the number of connected components in $f_1, f_2$, and see that this is sufficient to ensure estimability:

```r
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:

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

Then we analyse the largest WW-component:

```r
wwcomp <- compfactor(est$fe, WW = TRUE)
nlevels(wwcomp)
## [1] 933
wwset <- wwcomp == 1
sum(wwset)
## [1] 3129
summary(wwest <- felm(y ~ x | f1 + f2 + f3,
data = dataset, subset = wwset, exactDOF = TRUE))
```

```
Call:
  felm(formula = y ~ x | f1 + f2 + f3, data = dataset, exactDOF = TRUE, subset = wwset)

Residuals:
   Min     1Q Median     3Q    Max
-1.3765 -0.2784  0.0000  0.2791  1.5951

Coefficients:
  Estimate Std. Error  t value Pr(>|t|)
  x   0.994390   0.009889 100.696  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4858 on 2314 degrees of freedom
Multiple R-squared(full model): 0.9182  Adjusted R-squared: 0.8894
Multiple R-squared(proj model): 0.8138  Adjusted R-squared: 0.7483
F-statistic(full model): 31.91 on 814 and 2314 DF, p-value: < 2.2e-16
F-statistic(proj model): 1.011e+04 on 1 and 2314 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:

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

but a smaller number of identifiable coefficients.

```r
nrow(wwalpha <- getfe(wwest))
## [1] 816
```

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

```r
head(wwalpha)
## effect obs comp fe idx
## f1.35 1.9324241 1 1 f1 35
## f1.38 0.8049655 3 1 f1 38
## f1.40 0.2392413 3 1 f1 40
## f1.41 1.0896624 2 1 f1 41
## f1.42 0.6428771 4 1 f1 42
## f1.43 1.4268411 4 1 f1 43
```

```r
alpha[c(35, 38, 40:43),]
## effect obs comp fe idx
## f1.35 0.9581560 10 1 f1 35
## f1.38 0.6367389 9 1 f1 38
## f1.40 0.8802631 12 1 f1 40
## f1.41 0.8586243 13 1 f1 41
## f1.42 0.8983645 13 1 f1 42
## f1.43 1.2634715 12 1 f1 43
```

but there is no obvious relation between e.g. \( f1.35 - f1.36 \), they are very different in the two estimations. The coefficients are from different datasets, and the standard errors are large (\( \approx 0.7 \)) with this few observations for each factor level. The number of identified coefficients for each factor varies (these figures contain the two references):

```r
table(wwalpha[, "fe"])
##
## f1 f2 f3
## 417 198 201
```

References


