How lfe works

Simen Gaure

Abstract. Here is a proof for the demeaning method used in **lfe**, and a description of the methods used for solving the residual equations. As well as a toy-example.

1. Introduction

We assume we have an OLS model in matrix form

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

where X is a $(n \times k)$-matrix, and D is a $(n \times g)$-matrix. D is a set of dummies for e category variables. I.e. D is a block matrix, $D = \begin{bmatrix} D_1 & D_2 & \dots & D_e \end{bmatrix}$. That is, the entries of each D_i consists of 0 and 1, with only one non-zero entry per row. These are the dummies from a single factor, one column per level. Hence, the columns of each D_i are pairwise orthogonal. Though, in general, D_i is not orthogonal to D_j for $i \neq j$.

That is, in R the model will be

```
> Y ~ X1 + X2 + ... + Xk + D1 + D2 + ... + De
```

where `D1, D2, ..., De` are arbitrary factors. I.e. an entirely ordinary model which may easily be estimated by `lm`, or even with sparse-versions of the same.

g is the sum of the number of levels in the factors. Now, suppose $g \approx 10^6$, indeed, assume that all the factors have many levels, so that an unmanagable number of dummies will be created when we try to estimate, even if we sweep out the largest.

Then, we must do the math. Let's write the model in a slightly different block matrix form, to get hold of some facts of the Frisch-Waugh-Lovell theorem:

$$Y = \begin{bmatrix} X & D \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \end{bmatrix} + \epsilon$$

We get the normal equations

$$\begin{bmatrix} X & D \end{bmatrix}' \begin{bmatrix} X & D \end{bmatrix} \begin{bmatrix} \hat{\beta} \\ \hat{\alpha} \end{bmatrix} = \begin{bmatrix} X & D \end{bmatrix}' Y$$

which, when multiplied out, become

$$\begin{bmatrix} X'X & X'D \\ D'X & D'D \end{bmatrix} \begin{bmatrix} \hat{\beta} \\ \hat{\alpha} \end{bmatrix} = \begin{bmatrix} X' \\ D' \end{bmatrix} Y$$

We then write them as two rows

$$X'X\hat{\beta} + X'D\hat{\alpha} = X'Y \tag{2}$$

$$D'X\hat{\beta} + D'D\hat{\alpha} = D'Y, \tag{3}$$

and assume, for a moment, that we have removed sufficient reference levels from D, so that $D'D$ is invertible. Now, multiply through equation (3) with $X'D(D'D)^{-1}$ and subtract equation (2) from (3). This removes the $\hat{\alpha}$-term from (3). We then name $P = I - D(D'D)^{-1}D'$ to get

$$X'PX\hat{\beta} = X'PY.$$

Now, note that P is a projection, i.e. $P = P' = P^2$, hence we have $X'PX = X'P'PX = (PX)'PX$ and $X'PY = X'P'PY = (PX)'PY$ which yields

$$(PX)'PX\hat{\beta} = (PX)'PY \tag{4}$$

which is the normal equation of the system

$$PY = PX\beta + P\epsilon. \tag{5}$$

That is, $\hat{\beta}$ may be estimated from system (5), with the dummies removed, taking into account the adjusted degrees of freedom when computing the covariance matrix.

Moreover, by multiplying through equation (3) with $D(D'D)^{-1}$ and noting that $D(D'D)^{-1}D' = I - P$, we get

$$(I - P)X\hat{\beta} + D\hat{\alpha} = (I - P)Y \tag{6}$$

which may be reordered as

$$Y - (X\hat{\beta} + D\hat{\alpha}) = PY - PX\hat{\beta} \tag{7}$$

showing that the residuals of the projected system (5) equals the residuals of the original system (1).

All this is well-known as the Frisch-Waugh-Lovell theorem, and is not the main point here, that's why we're still in the "Introduction"-section.

2. What lfe does about this

The problem is to compute the projection P, so that we may estimate $\hat{\beta}$ from (5). Whenever $e = 1$, i.e. a single factor, applying P amounts to subtracting the group-means. This is known as the within-groups transformation, or centering on the means, or *demeaning*. But, what does it look like when we have more factors?

Here's the idea behind **lfe**, from [**3**]:

For each of the factors, we have a demeaning projection $P_i = I - D_i(D_i'D_i)^{-1}D_i'$. This is the projection onto the orthogonal complement of the range (column space) of D_i, called $R(D_i)^\perp$. These are easy to compute, it's just to subtract the means of each level. Similarly, P is the projection on $R(D)^\perp$. This one is not yet obvious how to compute.

There is a relation between all these range-spaces:

$$R(D)^\perp = R(D_1)^\perp \cap R(D_2)^\perp \cap \cdots \cap R(D_e)^\perp.$$

To see this, consider a vector $v \in R(D)^\perp$. By definition, it's orthogonal to every column in D, hence to every column in every D_i, thus v is in the intersection on the right hand side. Conversely, take a v which is in all the spaces $R(D_i)^\perp$. It's

orthogonal to every column of every D_i, hence it's orthogonal to every column in D, so it's in $R(D)^{\perp}$.

This relation may be written in terms of projections:

$$P = P_1 \wedge P_2 \wedge \cdots \wedge P_e.$$

Now, there's a theorem about projections [**4**, Theorem 1] which states that for every vector v, we have

$$Pv = \lim_{n\to\infty} (P_1 P_2 \cdots P_e)^n v. \tag{8}$$

In R, this looks like (with convergence to a tolerance `eps`):

```
> Pv <- v
> oldv <- v - 1
> fl <- list(D1, D2, ..., De)
> while (sqrt(sum((Pv - oldv)^2)) >= eps) {
+     oldv <- Pv
+     for (f in fl) Pv <- Pv - tapply(Pv, f, mean)[f]
+ }
```

So, there's how to compute Pv for an arbitrary vector v, just demean it with each projection in succession, over and over, until it gives up. We do this for every column of X and Y to find PY and PX, and then we may solve $\hat{\beta}$ from (4). This procedure has been wrapped up with a threaded C-routine in the function `felm`. Thus, the `X1,X2,...Xk` can be estimated efficiently by

```
> felm(Y ~ X1 + X2 + ... + Xk + G(D1) + G(D2) + ... + G(De))
```

If there is only one factor (i.e. $e = 1$), this reduces to the within-groups model.

3. The dummies?

To find $\hat{\alpha}$, the coefficents of all the dummies, we may write (6) as

$$D\hat{\alpha} = (Y - X\hat{\beta}) - (PY - PX\hat{\beta})$$

where the right hand side is readily computed when we know $\hat{\beta}$. There will be no more than e non-zeros in each row of D. This type of sparse system lends itself to solving by the Kaczmarz method ([**5**]).

The Kaczmarz method may be viewed as a variant of Halperin's Method of Alternating Projections, specifically for solving linear equations. The idea is that in a matrix equation like

$$Dx = b$$

we may view each row of the system $\langle d_i, x\rangle = b_i$ as an equation defining a hyperplane Q_i (where d_i is the i'th row of D). The solution set of the system is the intersection of all the hyperplanes $Q = Q_1 \cap Q_2 \cap \cdots \cap Q_n$. Thus, again, if the projection onto each Q_i is easy to compute (it is), we may use (8) on these projections to find a solution, starting from the zero-vector.

In our case, each row of the matrix D has exactly e non-zero entries, and they are equal to unity. This makes the computation of the projection on each Q_i easy and fast. We don't have to care about rank-deficiency (*you* do, if you're going to interpret the results); but we do remove consecutive duplicate rows.

Anyway, the Kaczmarz method converges to a solution $\hat{\alpha}$. Since we use 0 as our starting point, we compute the projection of the zero-vector onto the solution space, this is, by definition, the solution with minimal norm. From the Kaczmarz

method we don't get any indication of the rank-deficiency. (Though for $e = 2$, this can be inferred from the component-structure returned by `getfe`.) The method requires little memory, and it's way faster then most other methods.

A drawback is that the Kaczmarz method is not immediately parallelizable (though there's a variant by Cimmino which is, each iteration projects the point onto each hyperplane, then the next approximation is the centroid of these projections), and it does not yield any covariance matrix or standard errors. This is the default method used by `getfe`.

Alternatively, one may choose a sparse Cholesky solver. That is, we have from (3) that

$$D'D\hat{\alpha} = D'(Y - X\hat{\beta}).$$

In the case $e = 1$, we have that $D'D$ is diagonal, this is the within-groups case, and $\hat{\alpha}$ is just the group-means of the residuals $Y - X\hat{\beta}$. In the general case, we have a large, but sparse, linear system. This may be solved with the methods in package **Matrix**. This procedure has been packaged in the function `getfe`.

Now, it turns out that identification, hence interpretation, of the coefficients, *may* be a complicated affair. The reason is that the matrix $D'D$ may be rank-deficient in unexpected ways. It's sometimes not sufficient to remove a reference-level in each factor. In the case $e = 2$ these difficulties are well understood and treated in [**1**] and [**2**], as well as implemented in **lfe**. For larger e, this problem is harder, **lfe** uses a pivoted Cholesky-method to find linear dependencies in $D'D$, and removes them, but the resulting interpretation of the coefficients are in general not well understood. (This, of course, applies to the Kaczmarz method too).

4. An example

First we create a couple of covariates:

```
> set.seed(41)
> x <- rnorm(500)
> x2 <- rnorm(length(x))
> x3 <- rnorm(length(x))
```

Then we create some random factors, not too many levels, just for illustration, and some effects:

```
> f1 <- factor(sample(7, length(x), replace = TRUE))
> f2 <- factor(sample(4, length(x), replace = TRUE))
> f3 <- factor(sample(3, length(x), replace = TRUE))
> eff1 <- rnorm(nlevels(f1))
> eff2 <- rexp(nlevels(f2))
> eff3 <- runif(nlevels(f3))
```

Then we create an outcome with some normal residuals:

```
> y <- x + 0.5 * x2 + 0.25 * x3 + eff1[f1] + eff2[f2] + eff3[f3] +
+     rnorm(length(x))
```

Now, for illustration, create a demeaning function according to (8):

```
> demean <- function(v, fl) {
+     Pv <- v
+     oldv <- v - 1
+     while (sqrt(sum((Pv - oldv)^2)) >= 1e-07) {
+         oldv <- Pv
```

```
+         for (f in fl) Pv <- Pv - tapply(Pv, f, mean)[f]
+     }
+     Pv
+ }
```

and demean things

```
> fl <- list(f1, f2, f3)
> Py <- demean(y, fl)
> Px <- demean(x, fl)
> Px2 <- demean(x2, fl)
> Px3 <- demean(x3, fl)
```

And then we estimate it

```
> summary(lm(Py ~ Px + Px2 + Px3 - 1))

Call:
lm(formula = Py ~ Px + Px2 + Px3 - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7367 -0.6249 -0.0114  0.7014  3.0506

Coefficients:
    Estimate Std. Error t value Pr(>|t|)
Px   1.06543    0.04484  23.761  < 2e-16 ***
Px2  0.50988    0.04541  11.228  < 2e-16 ***
Px3  0.22739    0.04346   5.232 2.48e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.991 on 497 degrees of freedom
Multiple R-squared: 0.586,        Adjusted R-squared: 0.5835
F-statistic: 234.5 on 3 and 497 DF,  p-value: < 2.2e-16
```

Note that `lm` believes there are too many degrees of freedom, so the standard errors are too small.

The function `felm` in package **lfe** adjusts for the degrees of freedom, so that we get the same standard errors as if we had included all the dummies:

```
> summary(est <- felm(y ~ x + x2 + x3 + G(f1) + G(f2) + G(f3)))

Call:
   felm(formula = y ~ x + x2 + x3 + G(f1) + G(f2) + G(f3))

     Min       1Q   Median       3Q      Max
-2.73674 -0.62486 -0.01140  0.70142  3.05056

Coefficients:
   Estimate Std. Error t value Pr(>|t|)
x   1.06543    0.04539  23.472  < 2e-16 ***
x2  0.50988    0.04597  11.092  < 2e-16 ***
x3  0.22739    0.04400   5.168 3.46e-07 ***
---
```

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

Residual standard error: 1.003 on 485 degrees of freedom
Multiple R-squared: 0.9271   Adjusted R-squared: 0.9249
F-statistic: 411.4 on 15 and 485 DF, p-value: < 2.2e-16
*** Standard errors may be slightly too high due to more than 2 groups
```

We also illustrate how to fetch the group coefficients. We adjust the estimates to be comparable with the ones from `lm`.

```
> alpha <- getfe(est)
> r2 <- alpha["f2.1", "effect"]
> r3 <- alpha["f3.1", "effect"]
> alpha[1:7, "effect"] <- alpha[1:7, "effect"] + r2 + r3
> alpha[8:11, "effect"] <- alpha[8:11, "effect"] - r2
> alpha[12:14, "effect"] <- alpha[12:14, "effect"] - r3
> print(alpha, digits = 5)
         effect obs comp fe idx
f1.1  3.7660274  67    1 f1   1
f1.2  2.1057235  65    1 f1   2
f1.3 -0.7916539  70    1 f1   3
f1.4  3.9051240  65    1 f1   4
f1.5  0.0034456  85    1 f1   5
f1.6  2.6331931  78    1 f1   6
f1.7  2.4589590  70    1 f1   7
f2.1  0.0000000 140    1 f2   1
f2.2  1.2719890 120    1 f2   2
f2.3  0.1704565 114    1 f2   3
f2.4  2.0841700 126    1 f2   4
f3.1  0.0000000 151    1 f3   1
f3.2 -0.1576456 173    1 f3   2
f3.3 -0.2215718 176    1 f3   3
```

Here's the same estimation in `lm`, with dummies:

```
> summary(lm(y ~ x + x2 + x3 + f1 + f2 + f3 - 1))

Call:
lm(formula = y ~ x + x2 + x3 + f1 + f2 + f3 - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7367 -0.6249 -0.0114  0.7014  3.0506

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
x    1.065433   0.045392  23.472  < 2e-16 ***
x2   0.509879   0.045968  11.092  < 2e-16 ***
x3   0.227387   0.043999   5.168 3.46e-07 ***
f11  3.766027   0.155905  24.156  < 2e-16 ***
f12  2.105724   0.162775  12.936  < 2e-16 ***
f13 -0.791654   0.157569  -5.024 7.12e-07 ***
```

```
f14  3.905124   0.165229  23.635  < 2e-16 ***
f15  0.003446   0.148566   0.023   0.9815
f16  2.633193   0.151044  17.433  < 2e-16 ***
f17  2.458959   0.157898  15.573  < 2e-16 ***
f22  1.271989   0.127112  10.007  < 2e-16 ***
f23  0.170457   0.127707   1.335   0.1826
f24  2.084170   0.124974  16.677  < 2e-16 ***
f32 -0.157646   0.112902  -1.396   0.1633
f33 -0.221572   0.113139  -1.958   0.0508 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.003 on 485 degrees of freedom
Multiple R-squared: 0.9271,        Adjusted R-squared: 0.9249
F-statistic: 411.4 on 15 and 485 DF,  p-value: < 2.2e-16
```

References

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

[2] M. Andrews, L. Gill, T. Schank, and R. Upward, *High wage workers and low wage firms: negative assortative matching or limited mobility bias?*, J.R. Stat. Soc.(A) **171(3)** (2008), 673–697.

[3] S. Gaure, *OLS with Multiple High Dimensional Category Dummies*, (to appear) (2011).

[4] I. Halperin, *The Product of Projection Operators*, Acta Sci. Math. (Szeged) **23** (1962), 96–99.

[5] Kaczmarz, A. (1937). Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin International de l'Academie Polonaise des Sciences et des Lettres 35*, 355–357.

The Ragnar Frisch Centre for Economic Research, Oslo, Norway