lfe: Linear Regression With Many Fixed Effects

Günter J. Hitsch

1/11/2017

Why use lfe?

The lfe package estimates linear regression models with a large number of fixed effects

In the standard `lm` function we can estimate fixed effects as dummy variables. For example, if `id` is the identifier of a store and we want to estimate the effect of `x` on `y` while controlling for store fixed effects, then we estimate the regression:

```
fit = lm(y ~ x + factor(id), data = DT)
```

However, for a large number of fixed effects the `lm` function becomes inefficient and slow and eventually breaks down. As a rule of thumb, I would not use `lm` with more than about 500 fixed effects.

lfe is a great alternative to `lm`. It is designed to efficiently estimate regression models even with tens of thousands or more fixed effects.

Documentation

I am not aware of any good tutorial but usage of the package is straightforward. The official documentation is here:

https://cran.r-project.org/web/packages/lfe/lfe.pdf

Usage example

Let's simulate a data set with price and quantity observations in a panel data set with many stores and weeks. Experiment with the settings for the number of stores and weeks and check how quickly the lfm package can obtain estimates!

```
library(data.table)
library(lfe)

# Settings: Number of stores and weeks
N_stores = 200
N_weeks  = 100

# Settings: Regression coefficients, fixed effects, and error term standard deviation
price_effect  = -2.5
store_effects = 30 + 4*round(sin(1:N_stores),1)
week_effects  = round((0:(N_weeks-1))*(2/(N_weeks-1)), 1)
error_sd      = 1.5

# Total number of observations
N = N_stores*N_weeks

set.seed(1776)
DT = data.table(store_id = as.integer(rep(1:N_stores, each = N_weeks)),
                week     = as.integer(rep(1:N_weeks, times = N_stores)),
```

```r
                price    = round(runif(N, min = 4, max = 10), digits = 1)
               )

# Purchase quantity simulation
DT[, quantity := price_effect*price + store_effects[store_id] + week_effects[week]
                 + round(rnorm(N, sd = error_sd), 1)]
```

```r
head(DT)
```

```
   store_id week price quantity
1:        1    1   4.6    22.10
2:        1    2   5.0    21.30
3:        1    3   9.0    13.90
4:        1    4   4.2    23.40
5:        1    5   9.9     8.35
6:        1    6   6.3    18.65
```

Estimation

Now estimate the regression model using the `felm` (fixed-effects linear model) function:

```r
fit = felm(quantity ~ price | store_id + week, data = DT)
```

Note the syntax to add both store and week fixed effects to the regression formula.

The regression output will not include (fortunately!) the fixed effect estimates:

```r
summary(fit)
```

```

Call:
   felm(formula = quantity ~ price | store_id + week, data = DT)

Residuals:
    Min      1Q  Median      3Q     Max
-6.5398 -0.9856 -0.0091  1.0005  6.1748

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
price -2.507769   0.006148  -407.9   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.502 on 19700 degrees of freedom
Multiple R-squared(full model): 0.9249   Adjusted R-squared: 0.9238
Multiple R-squared(proj model): 0.8941   Adjusted R-squared: 0.8925
F-statistic(full model):  812 on 299 and 19700 DF, p-value: < 2.2e-16
F-statistic(proj model): 1.664e+05 on 1 and 19700 DF, p-value: < 2.2e-16
```

Obtain the fixed effects estimates

If you need to obtain the fixed effect estimates, use `getfe`. The results are stored in a data frame.

```r
FE = getfe(fit, se = TRUE)

head(FE)
```

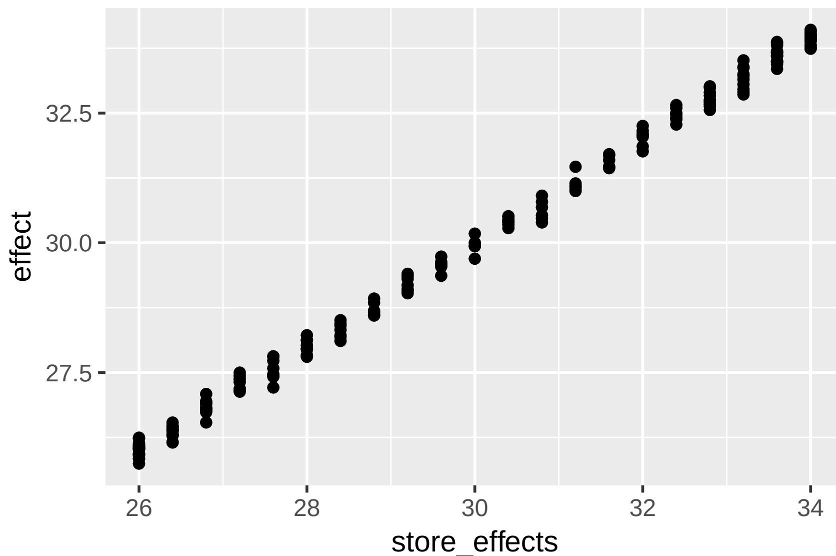
```
             effect obs comp       fe idx        se
store_id.1 33.24754 100    1 store_id   1 0.1962185
store_id.2 33.49384 100    1 store_id   2 0.1863863
store_id.3 30.34615 100    1 store_id   3 0.1888150
store_id.4 26.80839 100    1 store_id   4 0.1921345
store_id.5 26.03493 100    1 store_id   5 0.1943710
store_id.6 28.59850 100    1 store_id   6 0.1897269
```

Plot the estimated vs. the true fixed effects

True and estimated store fixed effects:

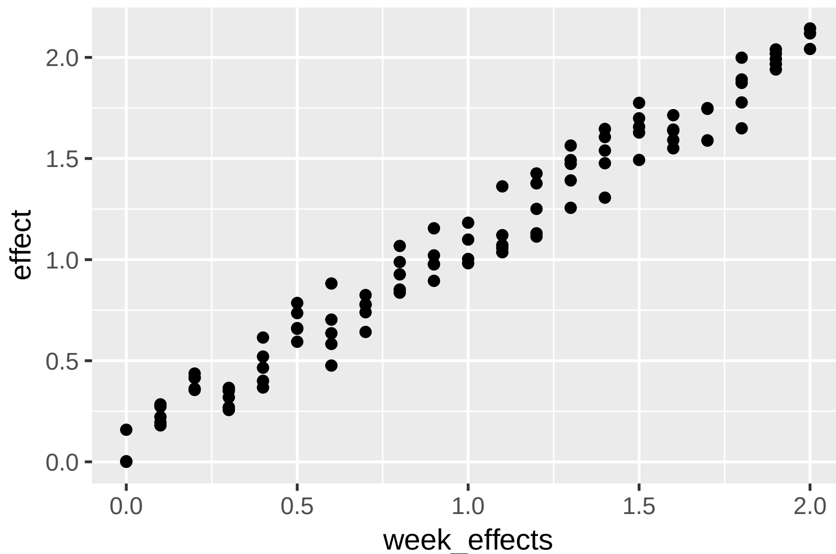
```r
library(ggplot2)
setDT(FE)           # Convert FE to a data.table

ggplot(FE[fe == "store_id"], aes(x = store_effects, y = effect)) +
   geom_point()
```



True and estimated week fixed effects:

```r
ggplot(FE[fe == "week"], aes(x = week_effects, y = effect)) +
   geom_point()
```



Prediction

The lfe package does not contain a `predict` method that works with `felm`. Hence we add the function `predict.felm` below.

```
predict.felm <- function(fit, newdata) {

   if (class(fit) != "felm") stop("'fit' is not a felm object")
   if (!("data.frame" %in% class(newdata))) stop("'newdata' must be a data.frame or data.table")

   setDT(newdata)

   # Predict output based on estimated coefficients, not inclucing fixed effects
   var_names = rownames(fit$coefficients)
   original_formula = paste("~ 0 +", paste(var_names, collapse = " + "))
   X = model.matrix(formula(original_formula), newdata)
   y = X %*% fit$coefficients

   # Add fixed effect values to prediction
   FE = as.data.table(getfe(fit))
   cols = c("fe", "idx")
   FE[, (cols) := lapply(.SD, as.character), .SDcols = cols]

   for (name in unique(FE$fe)) {
      fe_DT = newdata[, name, with = FALSE]
      fe_DT[, obs_no := .I]
      setnames(fe_DT, name, "idx")
      fe_DT[, idx := as.character(idx)]

      fe_DT = merge(fe_DT, FE[fe == name, .(idx, effect)], by = "idx")
      fe_DT = fe_DT[order(obs_no)]

      y = y + fe_DT$effect
   }

   return(y)
```

```
}
```

Let's test it:

```
predicted_quantity = predict.felm(fit, DT)
cor(DT$quantity, predicted_quantity)
```

```
      quantity
[1,] 0.9617414
```

If we increase prices by 3:

```
new_DT = copy(DT)
new_DT[, price := price + 3]
new_predicted_quantity = predict.felm(fit, new_DT)
```

```
mean(predicted_quantity)
```

```
[1] 13.50634
```

```
mean(new_predicted_quantity)
```

```
[1] 5.983033
```