MLE - Lab 12

Andy Ballard

April 21, 2017

Today

- Grades, end of semester housekeeping
- PLM
- Out of Sample Prediction

Housekeeping

You'll get your paper 2 grades and Homework 5 grades this weekend, at which point you'll have all your grades for the semester, save your final paper. Your final paper is due two weeks from today, Friday, May 1st at 1pm. I will hold office hours as usual (or with notice of changes) until then.

For the final paper: please format it as a paper. These labs are in an informal style that is suitable for class, but NOT for a paper. That is, your papers should have prose, and not just text comments in between snippets of code. If you do use an .Rmd to write your papers, please turn in both the .Rmd file and a .pdf, where the code in the .pdf is hidden. If you use this format, please be sure to format tables and figures properly, so that the axes and labels are easy to read and not just the names of variable objects in `R`.

Linear models with panel data

We'll do some hands on examples with panel data using the data Daniel talked about during lecture this week.

We have data from Ziliak (1997) about how labor supply reacts to wages. It is a balanced panel of 532 men from 1979-1988 (N=5320, T=10).

Again, T is the number of waves in a panel. So we have 10 waves for 532 individuals. $532 * 10 = 5320$.

The model we looked at in class was exceedingly simple, just logged annual hours worked regressed on logged annual wages:

$$lnhr_{it} = \beta * lnwg_{it} + \alpha_i + \epsilon_{it}$$

α_i signifies intercepts that vary over indivdiuals (but not over time). We could easily estimate a model where t he intercepts vary over individuals and time (α_{it}) which would be the most flexible version of the model. Here's what the data look like.

```
load(paste0(labPath, "hrs_wages.RData"))
head(data)
```

```
##   lnhr lnwg kids ageh agesq disab id year dyear1 dyear2 dyear3 dyear4
## 1 7.58 1.91    2   27   729     0  1 1979      1      0      0      0
## 2 7.75 1.89    2   28   784     0  1 1980      0      1      0      0
## 3 7.65 1.91    2   29   841     0  1 1981      0      0      1      0
## 4 7.47 1.89    2   30   900     0  1 1982      0      0      0      1
## 5 7.50 1.94    2   31   961     0  1 1983      0      0      0      0
## 6 7.50 1.93    2   32  1024     0  1 1984      0      0      0      0
```

```
##   dyear5 dyear6 dyear7 dyear8 dyear9 dyear10
## 1      0      0      0      0      0       0
## 2      0      0      0      0      0       0
## 3      0      0      0      0      0       0
## 4      0      0      0      0      0       0
## 5      1      0      0      0      0       0
## 6      0      1      0      0      0       0
```

For simplicity's sake, we'll just look at the same model with a few added variables. We'll also consider the number of children each individual has (`kids`), their age (`ageh`) and whether they're disabled (`disab`). The model we'll look at is:

$$lnhr_{it} = \beta_1 * lnwg_{it} + \beta_2 * kids_{it} + \beta_3 * age_{it} + \beta_4 * disab_{it} + \alpha_i + \epsilon_{it}$$

But first, let's replicate the model from class.

```
m1 <- plm(lnhr ~ lnwg, index=c('id', 'year'), data=data, effect='individual', model='random')
summary(m1)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = lnhr ~ lnwg, data = data, effect = "individual",
##     model = "random", index = c("id", "year"))
##
## Balanced Panel: n=532, T=10, N=5320
##
## Effects:
##                    var std.dev share
## idiosyncratic 0.05419 0.23278 0.676
## individual    0.02600 0.16125 0.324
## theta:  0.5847
##
## Residuals :
##     Min.  1st Qu.   Median  3rd Qu.     Max.
## -4.32000 -0.06680  0.00288  0.08720  0.79300
##
## Coefficients :
##             Estimate Std. Error  t-value  Pr(>|t|)
## (Intercept) 7.346041   0.036392 201.8561 < 2.2e-16 ***
## lnwg        0.119332   0.013631   8.7543 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    293.02
## Residual Sum of Squares: 288.86
## R-Squared:      0.014206
## Adj. R-Squared: 0.014021
## F-statistic: 76.6383 on 1 and 5318 DF, p-value: < 2.22e-16
```

You can also specify the same model using a special structure from the `plm` package, the `pdata.frame`. Then we don't have to specify the ID variables in the model, they're already part of the data.

```
pdata <- pdata.frame(data, c("id", "year"))
summary(pdata)
```

```
##       lnhr             lnwg              kids            ageh
##  Min.   :2.770   Min.   :-0.260   Min.   :0.000   Min.   :22.00
##  1st Qu.:7.580   1st Qu.: 2.370   1st Qu.:1.000   1st Qu.:32.00
##  Median :7.650   Median : 2.640   Median :2.000   Median :38.00
##  Mean   :7.657   Mean   : 2.609   Mean   :1.556   Mean   :38.92
##  3rd Qu.:7.780   3rd Qu.: 2.860   3rd Qu.:2.000   3rd Qu.:45.00
##  Max.   :8.560   Max.   : 4.690   Max.   :6.000   Max.   :60.00
##
##      agesq          disab              id            year
##  Min.   : 484   Min.   :0.0000   1      :  10   1979   : 532
##  1st Qu.:1024   1st Qu.:0.0000   2      :  10   1980   : 532
##  Median :1444   Median :0.0000   3      :  10   1981   : 532
##  Mean   :1586   Mean   :0.0609   4      :  10   1982   : 532
##  3rd Qu.:2025   3rd Qu.:0.0000   5      :  10   1983   : 532
##  Max.   :3600   Max.   :1.0000   6      :  10   1984   : 532
##                                  (Other):5260   (Other):2128
##      dyear1        dyear2        dyear3        dyear4        dyear5
##  Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0
##  1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0
##  Median :0.0   Median :0.0   Median :0.0   Median :0.0   Median :0.0
##  Mean   :0.1   Mean   :0.1   Mean   :0.1   Mean   :0.1   Mean   :0.1
##  3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0
##  Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0
##
##      dyear6        dyear7        dyear8        dyear9       dyear10
##  Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0   Min.   :0.0
##  1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0
##  Median :0.0   Median :0.0   Median :0.0   Median :0.0   Median :0.0
##  Mean   :0.1   Mean   :0.1   Mean   :0.1   Mean   :0.1   Mean   :0.1
##  3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0   3rd Qu.:0.0
##  Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0   Max.   :1.0
##
```

```
m2 <- plm(lnhr ~ lnwg, data=pdata, model='random')
summary(m2)
```

```
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = lnhr ~ lnwg, data = pdata, model = "random")
##
## Balanced Panel: n=532, T=10, N=5320
##
## Effects:
##                    var std.dev share
## idiosyncratic 0.05419 0.23278 0.676
## individual    0.02600 0.16125 0.324
## theta:  0.5847
##
## Residuals :
##     Min.  1st Qu.   Median  3rd Qu.     Max.
## -4.32000 -0.06680  0.00288  0.08720  0.79300
##
## Coefficients :
```

```
##             Estimate Std. Error  t-value  Pr(>|t|)
## (Intercept) 7.346041   0.036392 201.8561 < 2.2e-16 ***
## lnwg        0.119332   0.013631   8.7543 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    293.02
## Residual Sum of Squares: 288.86
## R-Squared:      0.014206
## Adj. R-Squared: 0.014021
## F-statistic: 76.6383 on 1 and 5318 DF, p-value: < 2.22e-16
```

Out of Sample Prediction

I have some bad news. A lot of the models we've been doing are probably overfit. Plus, we can't tell how accurate they are at predicting.

What is overfitting? It's when a model is too complicated for the data you're running it on. An overfit model can have coefficients, standard errors, and model fit statistics that are misleading. We generally think of the data we have as being a sample from some larger population. So our data contains some noise and is not a perfect reflection of the process that generates the data for the overall population. If you drew another sample, it would have its own noise, and your overfit model would not likely fit the new data.

This is one reason we want our model to fit the current sample, but new samples as well. Anoter reason is that we want our model to be able to predict what may happen if we collect new data, or to guess what may happen in the future.

Both out of sample prediction and cross validation can tell us whether our model is overfit, and test its predictive accuracy. The basic procedure is to fit a model with some amount of your data (training set) while holding back a portion of your data at random (test set), then running the model from your training set on the test set and comparing the model performance.

To show you how to do these procedures, we'll use some data from an older lab, the Fish data.

```
load(paste0(labPath, 'fish.RData'))

# Divide Fish data into a training and test set
fish$rand <- sample(1:2, nrow(fish), replace=TRUE)
table(fish$rand)
```

```
##
##  1  2
## 77 72
```

```
train <- fish[fish$rand==1,]
test <- fish[fish$rand==2,]

# Run OLS on training set
dv <- 'fhrev'
ivs <- c('muslim', 'income', 'elf', 'growth', 'britcol', 'postcom', 'opec')
modForm <- formula(paste0( dv, ' ~ ', paste(ivs, collapse=' + ') ))
mod <- lm(modForm, data=train)
```

Now we have a model for our training set. Note that in order to split the data up, I used a random procedure that should approximately split the data in half. Also, because we're running multiple models with the same formula on different data, it may be a good idea to use this stacked model formula process that we've done a times.

How can we compare models? One good way is to calculate the root mean squared error of the predictions for our models. First we can calculate the in-sample RMSE. We call this in-sample because we're using the training set model and the training set data, so we're predicting with the data that we used to make the model.

```
# Calculate in-sample RMSE

# First pull out training set observations
trainSet <- data.matrix( cbind(1, train[,ivs]) )
preds <- trainSet %*% coef(mod)

# Function to calculate RMSE
rmse <- function(pred, actual){
    sqrt( mean( (pred-actual)^2 ) )
}

# In-sample RMSE
rmse(preds, train$fhrev)
```

```
## [1] 1.225656
```

Now we can calculate the out of sample RMSE. To do so, we'll calculate predictions with the test set using the model from the training set. We are essentially using the test set as a scenario, or out new data, for predicting.

```
# Calculate out of sample RMSE
testSet <- data.matrix( cbind( 1, test[,ivs] ) )
preds <- testSet %*% coef(mod)
rmse(preds, test$fhrev)
```

```
## [1] 1.253269
```

The RMSEs are pretty dang close to each other, which suggests that our model is perhaps not too overfit.

Another way to compare models is to look at what happens to the coefficients when we run the same model on different data. Note that we aren't using the coefficients from the same model to do prediction with different sets of data here.

```
mod1 <- lm(modForm, data=train)
mod2 <- lm(modForm, data=test)
mod1Coefs <- round(summary(mod1)$'coefficients',3)
mod2Coefs <- round(summary(mod2)$'coefficients',3)
mod1Coefs
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.111      0.956   0.116    0.908
## muslim        -1.127      0.388  -2.902    0.005
## income         1.474      0.250   5.906    0.000
## elf           -0.238      0.571  -0.417    0.678
## growth         0.013      0.070   0.179    0.858
## britcol       -0.322      0.399  -0.809    0.422
## postcom        0.022      0.470   0.047    0.962
## opec          -1.970      0.690  -2.857    0.006
```

```
mod2Coefs
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.097      0.963  -0.101    0.920
## muslim        -1.230      0.337  -3.652    0.001
```

```
## income         1.428      0.260   5.492    0.000
## elf           -0.426      0.532  -0.801    0.426
## growth         0.126      0.058   2.188    0.032
## britcol        0.907      0.361   2.512    0.015
## postcom        0.446      0.422   1.055    0.295
## opec          -0.981      0.599  -1.638    0.106
```

And we can plot the results

```
# Create a dataframe for the plot
ggData <- data.frame(
    rbind(
        cbind( rownames(mod1Coefs), mod1Coefs[,1:2], 1 ),
        cbind( rownames(mod2Coefs), mod2Coefs[,1:2], 2 )
        )
    )
```

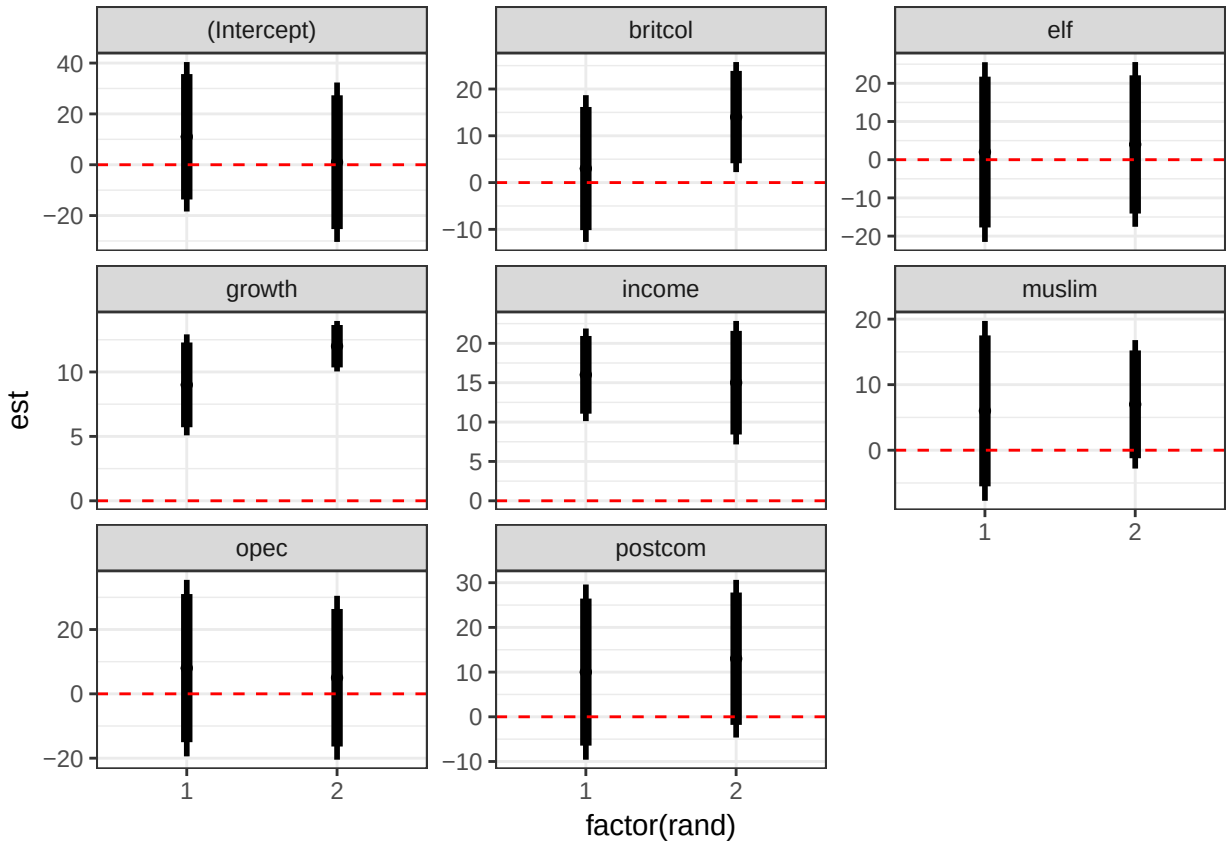
```
## Warning in data.row.names(row.names, rowsi, i): some row.names duplicated:
## 9,10,11,12,13,14,15,16 --> row.names NOT used
```

```
names(ggData) <- c('var', 'est', 'stderr', 'rand')

# Convert relev columns to numeric
for(ii in 2:3){ ggData[,ii] <- as.numeric(ggData[,ii]) }

# Lets get the 90 and 95 perc conf ints for the estimates
ggData$hi95 <- ggData$est + qnorm(.975)*ggData$stderr
ggData$lo95 <- ggData$est - qnorm(.975)*ggData$stderr
ggData$hi90 <- ggData$est + qnorm(.95)*ggData$stderr
ggData$lo90 <- ggData$est - qnorm(.95)*ggData$stderr

# Plot
tmp <- ggplot(ggData, aes(x=factor(rand), y=est))
tmp <- tmp + geom_point()
tmp <- tmp + geom_linerange(aes(ymin=lo95, ymax=hi95), lwd=1)
tmp <- tmp + geom_linerange(aes(ymin=lo90, ymax=hi90), lwd=2)
tmp <- tmp + facet_wrap(~var, scales='free_y')
tmp <- tmp + geom_hline(aes(yintercept=0), color='red', linetype=2)
tmp
```



So we've shown that if you cut the data sort of in half, it doesn't break our model. This is good news! But it still may be true that we've (although randomly) selected the data for our training and test sets in a way that is disproportionately favorable to our model. To get around this, we will create many training sets, or folds. We will run a model on each of the training sets, and then compare it to the reserved test set. We can use any number of folds, and we call this `k-fold cross-validation`.

For the sake of simplicity, we'll just use 4-fold cross-validation.

```
# Divide Fish data into k random subsets
k <- 4
fish$rand <- sample(1:k, nrow(fish), replace=TRUE)
table(fish$rand)
```

```
##
##  1  2  3  4
## 17 39 49 44
```

We have divided the data into 4 relatively equal partitions. You can subset based on samples in such a way that creates exactly the same number of observations in each fold if you'd like, but it's not important and may actually induce bias as the number of folds increases (and as the number of observations in each fold decreases). In practice, you rarely see more than 10-fold cross-validation.

Like the last example, we can compare the RMSEs for the different subsets. Although this time we won't use any prediction.

```
rmse <- function(pred, actual){ sqrt(mean( (pred-actual)^2 ))  }

coefCrossVal <- NULL
perf <- NULL
```

```
for(ii in 1:k){
  # subset into train and test
  train <- fish[fish$rand!=ii,]
  test <- fish[fish$rand==ii,]

  # get coefficients
  trainRes <- cbind(summary( lm(modForm, data=train)  )$'coefficients'[,1:2], ii)
  coefCrossVal <- rbind(coefCrossVal, trainRes)

  # get performance
  preds <- trainRes[,1] %*% t(data.matrix(cbind(1,test[,ivs])))
  perf <- c( perf, rmse(preds, test$fhrev) )
}

# Look at perf differences
perf
```

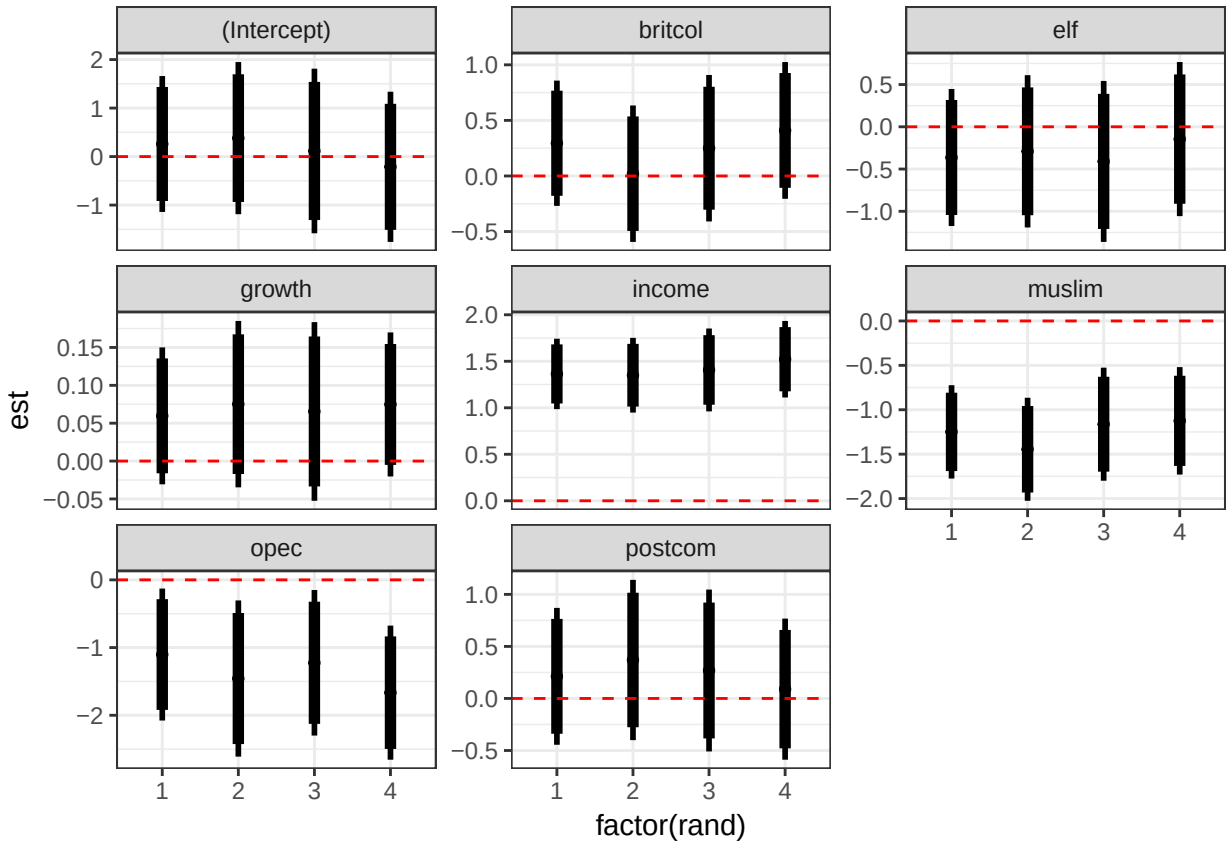
```
## [1] 1.259205 1.232766 1.094480 1.400491
```

We can plot our RMSEs for the training sets as well.

```
# organize our data
ggData <- data.frame( rownames(coefCrossVal), coefCrossVal, row.names=NULL  )
colnames(ggData) <- c('var', 'est', 'stderr', 'rand')

# Plot coefficient estimates
# Lets get the 90 and 95 perc conf ints for the estimates
ggData$hi95 <- ggData$est + qnorm(.975)*ggData$stderr
ggData$lo95 <- ggData$est - qnorm(.975)*ggData$stderr
ggData$hi90 <- ggData$est + qnorm(.95)*ggData$stderr
ggData$lo90 <- ggData$est - qnorm(.95)*ggData$stderr

# Plot
tmp <- ggplot(ggData, aes(x=factor(rand), y=est))
tmp <- tmp + geom_point()
tmp <- tmp + geom_linerange(aes(ymin=lo95, ymax=hi95), lwd=1)
tmp <- tmp + geom_linerange(aes(ymin=lo90, ymax=hi90), lwd=2)
tmp <- tmp + facet_wrap(~var, scales='free_y')
tmp <- tmp + geom_hline(aes(yintercept=0), color='red', linetype=2)
tmp
```




```
##################################################################
```

This is a reasonable comparison, running the same model on 4 different random subsets of the data and seeing what changes. This is a good way to test how fragile our model is as well. For instance, we can be reasonably certain that `income` is positively related to `fhrev`, and that `opec` and `muslim` are negatively related to `fhrev`, since the coefficients are significantly different than zero in the same direction in all the sets. These are the same three variables that were significant in the original model, so that's good news.

We could also look at the RMSE for a model trained on each of the training sets, and then predicted based on the remaining data.

Next week

- Sleeping in. Thanks for a great semester! I hope you learned things.