MLE: Lab 2

Shahryar Minhas

January 22, 2014

Setting up your workspace

```
# Start with a clean workspace
rm(list=ls())

# Function to load packages
loadPkg=function(toLoad){
  for(lib in toLoad){
    if(! lib %in% installed.packages()[,1])
      { install.packages(lib, repos='http://cran.rstudio.com/') }
    suppressMessages( library(lib, character.only=TRUE) ) }
}

# Load libraries
packs=c("ggplot2", 'ggthemes', 'MASS', 'arm')
loadPkg(packs)

# Set a theme for gg
theme_set(theme_bw())
theme_set(theme_economist())

# Functions that I use frequently
char = function(x){ as.character(x) }
num = function(x){ as.numeric(char(x)) }

# Relevant paths
lab2Path='~/Dropbox/Duke/Spring 2015/PS 733/lab2'
```

First, lets create a model object by running the crazy Muller & Seligson regression yet again.

```
# Load data
msrepPath=paste0(lab2Path, "/Msrepl87.asc")
msrep = read.table(msrepPath, header=TRUE)

# Create silly logged version of DV
msrep$deaths75ln = log(msrep$deaths75+1)

# Create logs of other things
msrep$deaths70ln = log(msrep$deaths70+1)
msrep$sanctions75ln = log(msrep$sanctions75+1)
msrep$sanctions70ln = log(msrep$sanctions70+1)
msrep$energypcln = log(msrep$energypc+1)

# Running a linear regression
ivs=c('upper20', 'energypcln', 'intensep',
```

```
      'sanctions70ln', 'sanctions75ln', 'deaths70ln')
olsForm=formula(
  paste0('deaths75ln ~ ',
         paste(ivs, collapse=' + ') )
  )
mod1 = lm(olsForm, data=msrep)

# View model results
summary(mod1)
```

Visualizing regression results

Tables suck, lets get a visualization of the coefficient results.

```
coefplot(mod1)
```

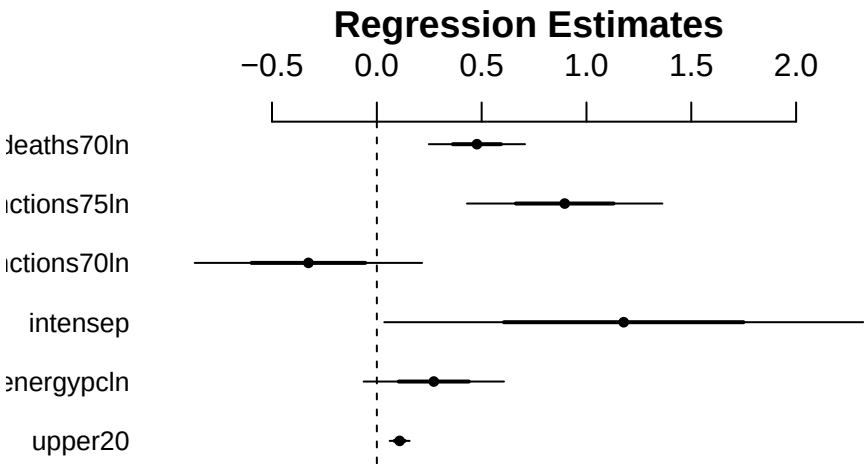

Figure 1: Ugly coefficient plot

Lets make a nicer looking coefficient plot. First we need to generate the data to plot. What data do we need?

```
# Pull out relevant info from model object
varName=rownames(summary(mod1)$coefficients)
varName=c('Intercept', 'Upper 20% Income Share, 1970',
          'Ln(Energy Cons.), 1970', 'Separatism, 1975',
          'Ln(Sanctions), 1968-72', 'Ln(Sanctions), 1973-77',
          'Ln(Deaths), 1968-72')
mean=summary(mod1)$coefficients[,1]
serror=summary(mod1)$coefficients[,2]

# Calculate confidence intervals around regression estimates
up95=mean+qnorm(0.975)*serror
lo95=mean-qnorm(0.975)*serror
up90=mean+qnorm(0.95)*serror
lo90=mean-qnorm(0.95)*serror

# Combine in dataframe
```

```
coefData=cbind(varName, mean, serror, up95, lo95, up90, lo90)
coefData=data.frame(coefData, row.names=NULL)

# Check class of variables in dataframe
class(coefData$up95)
```

```
## [1] "factor"
```

```
# Lets clean this up
convNumDcol=function(data, vars){
  for(var in vars){ data[,var]=num(data[,var]) }
    return( data ) }
coefData=convNumDcol(coefData, names(coefData)[2:length(coefData)])

# Lets check to make sure class is correct now
class(coefData$up95)
```

```
## [1] "numeric"
```

Now lets produce a nicer looking coefficient plot.

```
tmp=ggplot(data=coefData[-1,], aes(x=varName))
tmp=tmp + geom_linerange(aes(ymin=lo95, ymax=up95), size=.3)
tmp=tmp + geom_linerange(aes(ymin=lo90, ymax=up90), size=1)
tmp=tmp + geom_point(aes(y=mean))
tmp=tmp + geom_hline(yintercept=0, linetype=2, color = "red")
tmp=tmp + coord_flip() + xlab('') + ylab('')
tmp=tmp + theme(axis.ticks=element_blank())
tmp
```

Meaningfully interpreting marginal effects

Our primary concern should not be whether the variables we throw into a regression are significant. If you deal with datasets of a sufficient size finding significant relationships is trivial (and misleading).

What we actually care about are the substantive implications of the independent variables that we are testing. For example, in the case of Muller & Seligson's model, it is clear that if the upper 20% have a greater share of income, we can expect some uptick in political deaths, but by how much does this variable need to change for us to see an increase in political deaths?

There are many ways at getting this question. The simplest is to just start looking at marginal effects. Our regression model has the following linear form:

$$deaths75ln \;=\; \beta_0 + \beta_1 upper20 + \beta_2 energypcln + \ldots + \beta_6 deaths70ln$$

To determine the marginal effect of the upper20 variable, we simply take the partial derivative of deaths75ln with respect to upper20 and find:

$$\frac{\partial deaths75ln}{\partial upper20} \;=\; 0.1088$$

Figure 2: Economist coefficient plot

Thus a one unit change in upper20, holding all else constant, produces a 0.1088 point change in the number of logged deaths from political violence per 1m. That's great, but we still don't know if this is a meaningful change.

Lets go back to our data. A shift from the 25th to 75th percentile in upper20 is equal to 15. This translates to a 1.6316 change in our dependent variable.

How meaningful is this change? Well, here's a boxplot of our dependent variable, so you tell me.

```
tmp=ggplot(msrep, aes(x=factor(0), y=deaths75ln)) + geom_boxplot()
tmp=tmp + coord_flip() + xlab('') + scale_x_discrete(breaks=NULL)
tmp=tmp + ylab('Ln(Deaths from Political Violence per 1m), 1973-77')
tmp
```

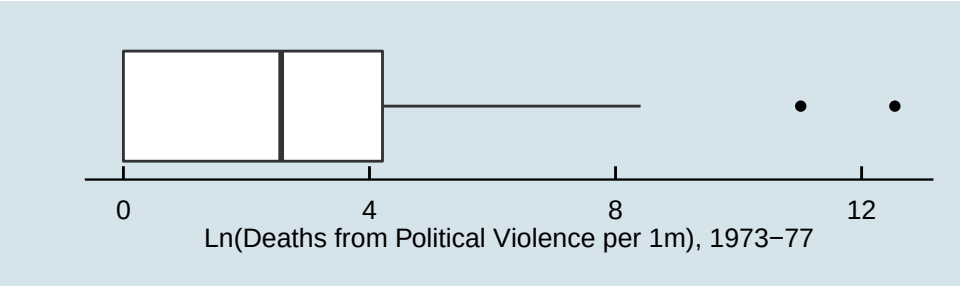

Figure 3: DV boxplot

Meaningfully interpreting coefficient estimates via simulations

What have we ignored in this discussion? Uncertainty. In making predictions about the effect of our independent variables we have completely ignored the standard errors from our coefficients. Additionally, we don't at all account for the uncertainty from the model itself.

We'll use simulations to take uncertainty into account. What are simulations...

```
set.seed(6886)
```

Monty Hall problem

```
sims = 1000     # number of simulations
doors = c(1,2,3)    # doors
WinNoSwitch = WinSwitch = 0 # these two will count the number of wins when switching and not switching.

for(i in 1:sims){               # the loop
    WinDoor = sample(doors,1)       # randomly pick a winning door
    choice = sample(doors,1)        # randomly pick a door
    # now one door is opened and only 2 remain:
    if(WinDoor==choice) DoorsLeft = c(WinDoor, sample(doors[doors!=WinDoor],1))
        # if you picked the right door initially, they will randomly open one of the others
    if(WinDoor!=choice) DoorsLeft = c(WinDoor, choice)
        # if you picked the wrong door initially, they will keep yours and the winning door

    # choice whether to switch or not
    noswitch = choice
    switch = DoorsLeft[DoorsLeft!=choice]

    # if you win, add 1 to the respective counter
    if(noswitch==WinDoor) WinNoSwitch = WinNoSwitch+1
    if(switch==WinDoor) WinSwitch = WinSwitch+1
}

paste("Probability | No Switch = ",WinNoSwitch/sims,"", sep="")
```

```
## [1] "Probability | No Switch = 0.334"
```

```
paste("Probability | Switch = ",WinSwitch/sims,"", sep="")
```

```
## [1] "Probability | Switch = 0.666"
```

Birthday problem

```
sims = 1000
people = 23
days = seq(1,365,1)
sameday = 0
```

```r
for(i in 1:sims){
  birthdays = sample(days, people, replace=T)
    if(length(unique(birthdays))<people) sameday = sameday+1
}

paste0("Probability at least 2 people with same birthday = ",sameday/sims,"")
```

```
## [1] "Probability at least 2 people with same birthday = 0.491"
```

Back to the point

Our regression results contain information about both the point estimates of our parameters and the standard errors of each of those parameters. We can use this information to generate predictions that take into account the uncertainty of our regression coefficients.

```r
# Scenario 1: High income inequality...other vars at central tendency
# Scenario 2: Low income inequality...other vars at central tendency
means=apply(msrep[,ivs], 2, function(x){ mean(x, na.rm=TRUE) })
minMaxSep=quantile(msrep[,ivs[1]], probs=c(0,1), na.rm=TRUE)

scens=rbind(c(1, minMaxSep[1], means[2:length(means)]),
              c(1, minMaxSep[2], means[2:length(means)]) )

# Simulate additional parameter estimates from multivariate normal
sims=1000
draws = mvrnorm(sims, coef(mod1), vcov(mod1))

# Get predicted values using matrix multiplication
preds = draws %*% t(scens)

#plotting sim results
plot(density(preds[,1]), col = "red", bty = "n",
     las = 1, xlim=c(-4, 10), lwd = 3, main='', ylim=c(0,1),
     xlab = "Logged Average Deaths per Million")
lines(density(preds[,2]), col = "blue", lwd = 3)
legend(x=-.4,y=.95,legend=c("Low upper20"),
       text.col=c("red"),bty="n", cex = 0.75)
legend(x=3.7,y=0.95,legend=c("High upper20"),
       text.col=c("blue"),bty="n", cex = 0.75)
```

We have taken into account the systematic component of the uncertainty in our regression model but we should also take into account fundamental uncertainty.

```r
# Model uncertainty
sigma = sqrt(sum(resid(mod1)^2)/df.residual(mod1))

# Generating expected values
exp = apply(preds, 2, function(x){
  rnorm(sims, x, sigma)
})

# Repeat code from before
```

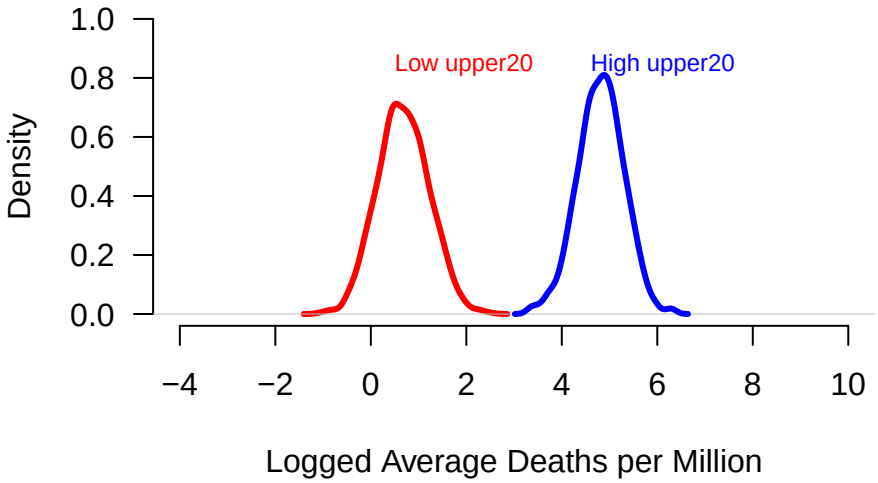

Figure 4: Simulation with Systematic Uncertainty

```
plot(density(preds[,1]), col = "red", bty = "n",
     las = 1, xlim=c(-4, 10), lwd = 3, main='', ylim=c(0,1),
     xlab = "Logged Average Deaths per Million")
lines(density(preds[,2]), col = "blue", lwd = 3)
legend(x=-.4,y=.95,legend=c("Low upper20"),
       text.col=c("red"),bty="n", cex = 0.75)
legend(x=3.7,y=0.95,legend=c("High upper20"),
       text.col=c("blue"),bty="n", cex = 0.75)
# Add lines with fundamental uncertainty
lines(density(exp[,1]), col='coral', lwd=3)
lines(density(exp[,2]), col='cornflowerblue', lwd=3)
```

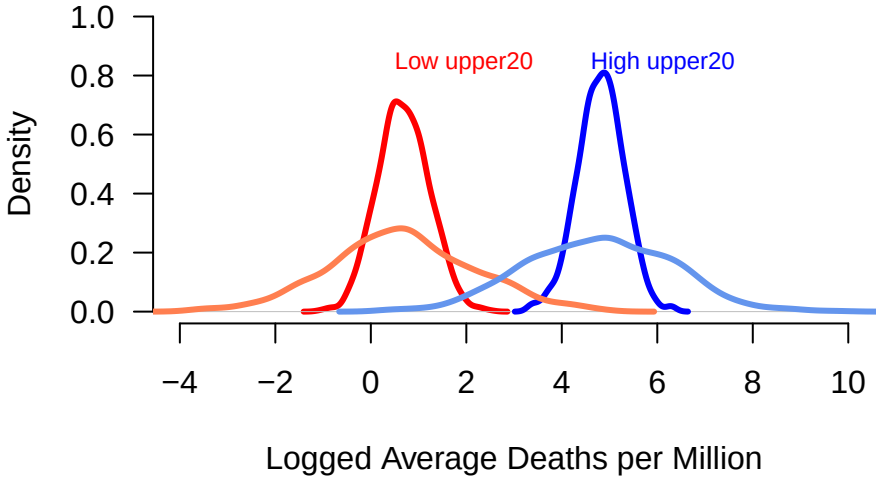

Figure 5: Simulation with Fundamental Uncertainty