MLE - Lab 11

Andy Ballard

March 30, 2017

Today

- Regression tables in LaTeX
- Selected Problems from Homework 4
- Hierarchical Models

Regression tables in LaTeX

```
d_2010 <- WDI(indicator = c("NY.GDP.PCAP.CD", "SP.DYN.IMRT.IN",
                             "SH.MED.PHYS.ZS"), start = 2010, end = 2010,
              extra = TRUE)
```

There are a lot of unwanted columns. We'll just keep `country`, `year`, and three variables of interest:`NY.GDP.PCAP.CD`, `SP.DYN.IMRT.IN`, `SH.MED.PHYS.ZS`.

```
d_2010 <- d_2010[d_2010$region != "Aggregates",
       c("country", "year", "NY.GDP.PCAP.CD", "SP.DYN.IMRT.IN",
         "SH.MED.PHYS.ZS")]

#Rename columns
colnames(d_2010)
```

```
## [1] "country"        "year"           "NY.GDP.PCAP.CD" "SP.DYN.IMRT.IN"
## [5] "SH.MED.PHYS.ZS"
```

```
colnames(d_2010)[3:5] <- c('gdppc', 'infant_mortality', 'number_of_physician')
colnames(d_2010)
```

```
## [1] "country"             "year"                "gdppc"
## [4] "infant_mortality"    "number_of_physician"
```

```
#Log GDP/capita
d_2010$log_gdppc <- log(d_2010$gdppc)

#Remove missing data
d_2010 <- na.omit(d_2010) #This would be a good candidate for imputation, FYI

#Build some linear models
m1 <- lm(infant_mortality ~ log_gdppc, data = d_2010)
summary(m1)
```

```
##
## Call:
## lm(formula = infant_mortality ~ log_gdppc, data = d_2010)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.241  -8.804  -0.650   6.651  50.710
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 134.3284     6.0152   22.33   <2e-16 ***
## log_gdppc   -12.7598     0.6921  -18.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.82 on 149 degrees of freedom
## Multiple R-squared:  0.6952, Adjusted R-squared:  0.6932
## F-statistic: 339.9 on 1 and 149 DF,  p-value: < 2.2e-16
```

```
m2 <- lm(infant_mortality ~ log_gdppc + number_of_physician, data = d_2010)
summary(m2)
```

```
##
## Call:
## lm(formula = infant_mortality ~ log_gdppc + number_of_physician,
##     data = d_2010)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.293  -7.913  -1.006   5.989  50.929
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         121.7595     7.3388  16.591  < 2e-16 ***
## log_gdppc           -10.7310     0.9802 -10.948  < 2e-16 ***
## number_of_physician  -2.6707     0.9343  -2.858  0.00487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.52 on 148 degrees of freedom
## Multiple R-squared:  0.7112, Adjusted R-squared:  0.7073
## F-statistic: 182.2 on 2 and 148 DF,  p-value: < 2.2e-16
```

We can report the model in a nice, journal-ready format. The `stargazer` library takes your model objects and generates tables in LaTeX.

```
# If using knir, use the option results='asis'
stargazer(m1, m2)
```

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Mar 31, 2017 - 1:17:17 AM

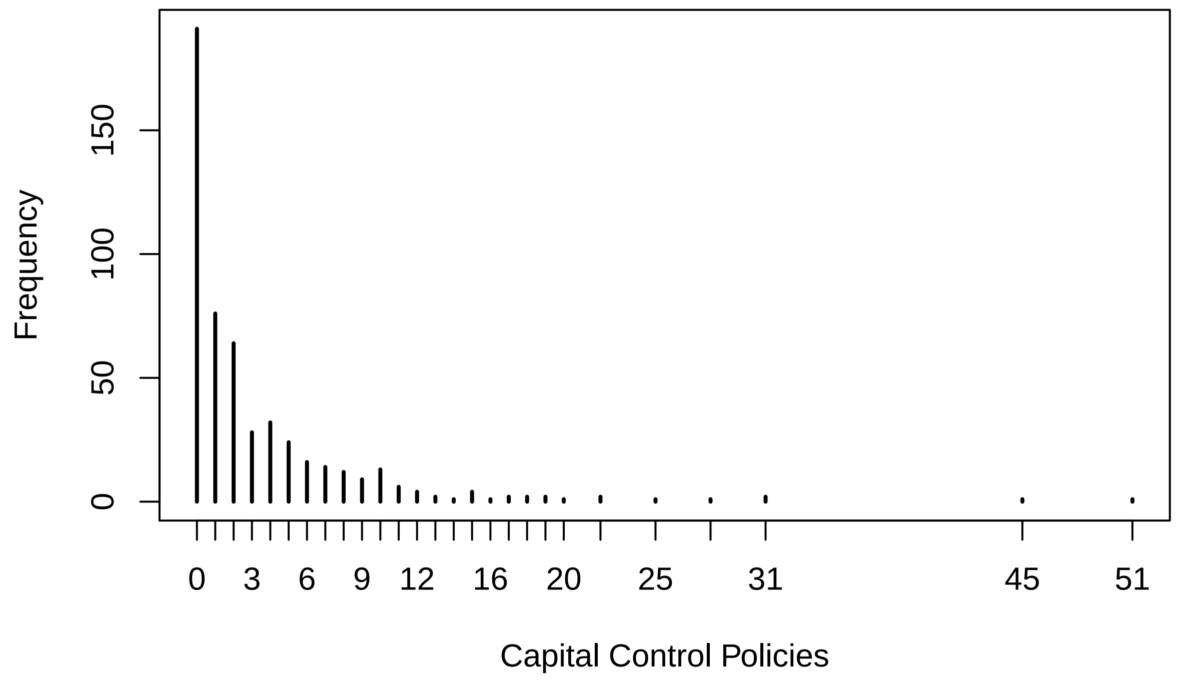
Homework 4

Plot frequency of DV, examine mean and variance

```
#Load data
load(paste0(labPath, "capitalcontrols.RData")); cap <- capitalcontrols; rm(capitalcontrols)

#Look at DV frequency
plot(table(cap$totalcha), ylab="Frequency", xlab="Capital Control Policies")
```

Table 1:

	Dependent variable:	
	infant_mortality	
	(1)	(2)
log_gdppc	-12.760^{***}	-10.731^{***}
	(0.692)	(0.980)
number_of_physician		-2.671^{***}
		(0.934)
Constant	134.328^{***}	121.759^{***}
	(6.015)	(7.339)
Observations	151	151
R^2	0.695	0.711
Adjusted R^2	0.693	0.707
Residual Std. Error	12.815 (df = 149)	12.518 (df = 148)
F Statistic	339.883^{***} (df = 1; 149)	182.204^{***} (df = 2; 148)
Note:		*p<0.1; **p<0.05; ***p<0.01



```
#Descriptives for DV
summary(cap$totalcha)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   3.223   4.000  51.000
```

```r
sum(cap$totalcha==0)/nrow(cap) #37% 0s
```

```
## [1] 0.3730469
```

```r
#Ratio of variance to mean of DV (Poisson assumes is 1)
var(cap$totalcha)/mean(cap$totalcha) #What does this suggest?
```

```
## [1] 8.928115
```

Modeling the number of policy changes per government as a count model

Let's look at the log-likelihood for a Poisson model.

The likelihood contribution of one observation for the Poisson distribution is:

$$p(Y = y_i|\mu) = \frac{e^{-\mu}\mu^{y_i}}{y_i!}$$

For n observations, we have the product of these likelihood contributions:

$$L(\mu|y) = \prod_{i=1}^{n} \frac{e^{-\mu}\mu^{y_i}}{y_i!}$$

We get the log-likelihood by taking the natural log of the above:

$$\ell(\mu|y) = -n\mu + ln(\mu)\sum_{i}^{n} y_i - \sum_{i}^{n} ln(y_i!)$$

Since we model the mean as $E(Y_i) = \mu = e^{X_i\beta}$, you can substitute this for μ. You could also specify that X_i is a vector of values for `veto players`, `international constraints`, and their interaction, and that y_i is the number of capital control policies per government.

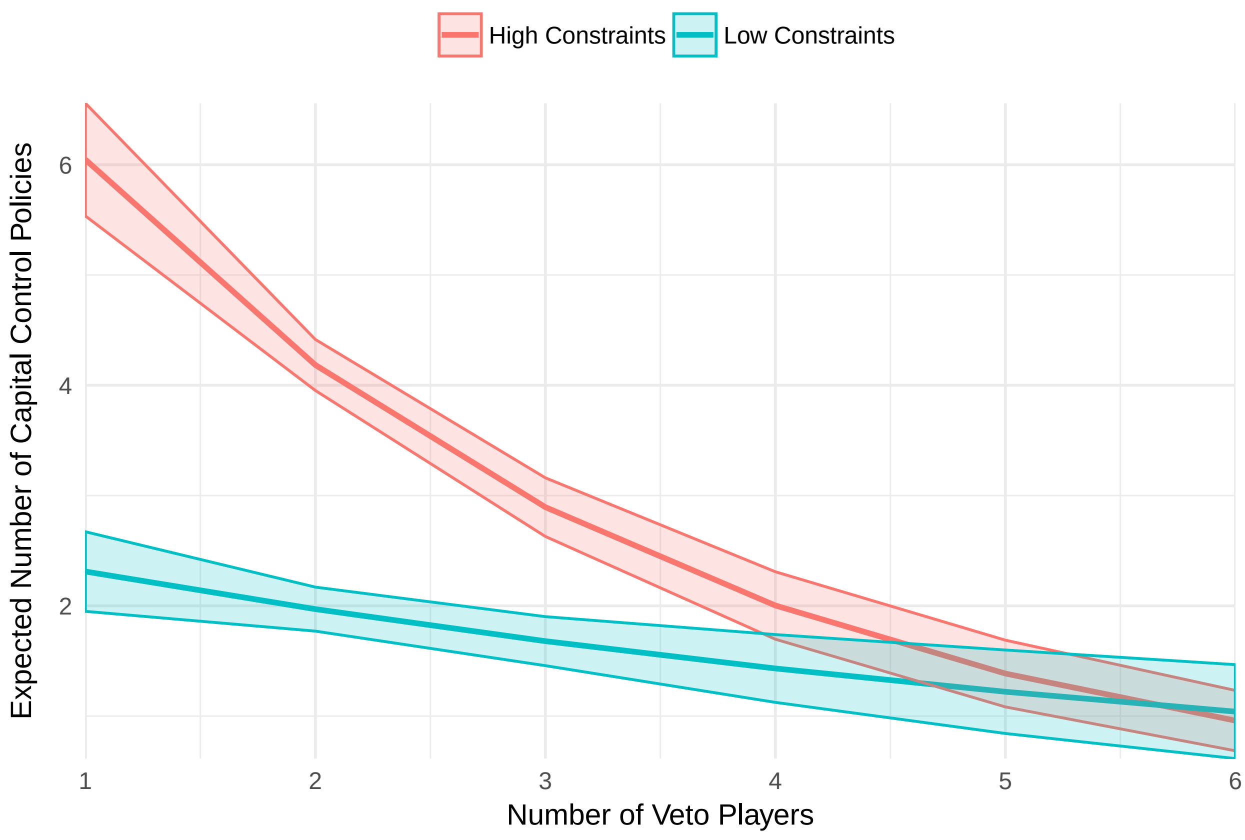
Now let's run a model.

```r
modForm <- formula("totalcha ~ envp + loconst + envp:loconst")

mp.full <- glm(modForm, data=cap, family=poisson)
```

To run a likeliehood ratio test, one way is to run another model with just the `envp` variable and use the `lrtest` function.

Now we can look at the predicted number of policy changes when international constraints are (or aren't) present and over the range of veto players.

```r
#Range for scenario varying veto players
s.range <- sort(cap$envp)

#Scenario 1: low constraints
scen1 <- data.frame(1, envp=s.range, loconst=1)
scen1.pred <- predict(mp.full, newdata=scen1, type = "response", se.fit=TRUE) %>% data.frame()
lo.1 <- scen1.pred$fit-1.96*scen1.pred$se.fit
hi.1 <- scen1.pred$fit+1.96*scen1.pred$se.fit
scen1.data <- data.frame(pred=scen1.pred$fit, lo=lo.1, hi=hi.1, s.range, constraint=as.factor(scen1$loco
#Scenario 2: high constraints
scen2 <- data.frame(1, envp=s.range, loconst=0)
scen2.pred <- predict(mp.full, newdata=scen2, type="response", se.fit=TRUE) %>% data.frame()
lo.2 <- scen2.pred$fit-1.96*scen2.pred$se.fit
hi.2 <- scen2.pred$fit+1.96*scen2.pred$se.fit
scen2.data <- data.frame(pred=scen2.pred$fit, lo=lo.2, hi=hi.2, s.range, constraint=as.factor(scen2$loco

#Combine and plot
scen.data <- data.frame(rbind(scen1.data, scen2.data))

scen.plot <- ggplot(scen.data, aes(x=s.range, y=pred, ymin=lo, ymax=hi, color=factor(constraint))) +
  geom_line(aes(colour=factor(constraint)),size=1) +
  geom_ribbon(aes(ymin=lo, ymax=hi, fill=factor(constraint)), alpha=0.2)+
  scale_y_continuous(name="Expected Number of Capital Control Policies",expand=c(0,0)) +
  scale_x_continuous(name="Number of Veto Players", expand=c(0,0)) +
  scale_fill_discrete(name="International Constraints", labels=c("High Constraints", "Low Constraints"))
  scale_colour_discrete(name="International Constraints", labels=c("High Constraints", "Low Constraints

scen.plot <- scen.plot + theme(legend.position='top', legend.title=element_blank(), axis.ticks=element_

scen.plot
```



What about a negative Binomial model? The data surely calls for one (via cursory look at mean and variance, and dispersion tests).

So does a negative Binomial model change anything? You can run a likelihood ratio test between our full Poisson model and the exact model formula using a negative Binomial model.

```r
mnb.full <- glm.nb(modForm, data=cap)

#Two tests, odTest or lrtest
odTest(mnb.full, alpha=0.05) #"overdispersion" test
```

```
## Likelihood ratio test of H0: Poisson, as restricted NB model:
## n.b., the distribution of the test-statistic under H0 is non-standard
## e.g., see help(odTest) for details/references
##
## Critical value of test statistic at the alpha= 0.05 level: 2.7055
## Chi-Square Test Statistic =  1453.4651 p-value = < 2.2e-16
```

```r
lrtest(mp.full, mnb.full)
```

```
## Likelihood ratio test
##
## Model 1: totalcha ~ envp + loconst + envp:loconst
## Model 2: totalcha ~ envp + loconst + envp:loconst
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   4 -1849.7
## 2   5 -1123.0  1 1453.5  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It does look like we need to use a negative Binomial model. The rest of the homework is just going through the same procedure as above, but for the NB model, and comparing the two. Some of the relationships will change.

Hierarchical Models

We gon' need some data. We'll use data from the American National Election Study.

Here are the variables we'll use:

`partyid7`: Party identification (Left-right, 7pt. Scale) `state`: State `age`: Age in years `female`: Female dummy `black`: Black dummy `year`: Year of survey `married`: Married dummy `educ`: Educational attainment `urban`: 1=urban, 2=suburban, 3=rural `union`: Union member dummy `south`: Southern state dummy

```r
load(paste0(labPath, "partyid.RData")); pid <- partyid; rm(partyid)
pid$union <- 2 - pid$union
```

Our DV for this tutorial will be party ID, which is on a 1 to 7 scale, with 1 being the strongest Democrats and 7 being the strongest Republicans. A natural hierarchy here is individuals within years. There are 15 iterations of the survey, every two years from 1972 to 2000.

Let's look at a fully pooled model (no fixed or random effects). Also, here's a good discussion of why hierarchical models might be a good idea (http://andrewgelman.com/2011/10/15/the-bias-variance-tradeoff/).

We'll use `urban`, `union`, `south`, `female`, `age`, `south`, and `black` as predictors.

```r
m.pooled <- lm(partyid7 ~ urban + union + south + female + age + south +
                 black, data=pid)
summary(m.pooled)
```

```
##
## Call:
## lm(formula = partyid7 ~ urban + union + south + female + age +
##     south + black, data = pid)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3774 -1.8385 -0.1853  1.8426  5.5868
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.153506   0.129110  32.170  < 2e-16 ***
## urban        0.099977   0.042703   2.341 0.019269 *
## union       -0.707641   0.079482  -8.903  < 2e-16 ***
## south       -0.181207   0.075191  -2.410 0.015999 *
## female      -0.210853   0.063349  -3.328 0.000881 ***
## age         -0.004003   0.001822  -2.198 0.028037 *
## black       -1.741557   0.108013 -16.124  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.986 on 3978 degrees of freedom
## Multiple R-squared:  0.09458,    Adjusted R-squared:  0.09322
## F-statistic: 69.26 on 6 and 3978 DF,  p-value: < 2.2e-16
```

Things are significant. That's good, it will allow us to see changes in the model. See any weird relationships?

What happens if we unpool the data and look at a model with years as constants?

```
m.unpooled <- lm(partyid7 ~ urban + union + south + female + age + south +
                 black + factor(year) - 1, data=pid)
summary(m.unpooled)
```

```
##
## Call:
## lm(formula = partyid7 ~ urban + union + south + female + age +
##     south + black + factor(year) - 1, data = pid)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5464 -1.7762 -0.1733  1.8143  5.7709
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## urban             0.106312   0.042739   2.487  0.01291 *
## union            -0.705609   0.079768  -8.846  < 2e-16 ***
## south            -0.178387   0.075436  -2.365  0.01809 *
## female           -0.207802   0.063393  -3.278  0.00105 **
## age              -0.004509   0.001834  -2.458  0.01401 *
## black            -1.746730   0.108229 -16.139  < 2e-16 ***
## factor(year)1972  4.177731   0.172643  24.199  < 2e-16 ***
## factor(year)1974  4.092666   0.183259  22.333  < 2e-16 ***
## factor(year)1976  4.349674   0.176918  24.586  < 2e-16 ***
## factor(year)1978  4.081320   0.164409  24.824  < 2e-16 ***
## factor(year)1980  4.316158   0.202120  21.354  < 2e-16 ***
## factor(year)1982  3.791020   0.194560  19.485  < 2e-16 ***
## factor(year)1984  4.284824   0.166414  25.748  < 2e-16 ***
## factor(year)1986  3.992247   0.161993  24.645  < 2e-16 ***
## factor(year)1988  4.349200   0.172366  25.232  < 2e-16 ***
## factor(year)1990  4.058188   0.165360  24.541  < 2e-16 ***
## factor(year)1992  4.131025   0.165405  24.975  < 2e-16 ***
## factor(year)1994  4.279889   0.166019  25.779  < 2e-16 ***
## factor(year)1996  4.246617   0.176266  24.092  < 2e-16 ***
## factor(year)1998  4.003317   0.181063  22.110  < 2e-16 ***
## factor(year)2000  4.439006   0.271913  16.325  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.984 on 3964 degrees of freedom
## Multiple R-squared:  0.7824, Adjusted R-squared:  0.7813
## F-statistic: 678.8 on 21 and 3964 DF,  p-value: < 2.2e-16
```

Nothing really changed, but we can see that we explained a ton more of the variance. Obviously, there is something about including the years that matters. Now let's allow the years to have slopes and not just be constants, using the `lme4` package.

First, we'll just use a simple model with no predictors, and only the year random effects, and we can compare it to a fixed effects model with only the year fixed effects.

```
m0 <- lmer(partyid7 ~ 1 + (1 | year), data=pid)
summary(m0)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: partyid7 ~ 1 + (1 | year)
##    Data: pid
##
## REML criterion at convergence: 17170.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.3154 -0.8269 -0.3048  1.0898  1.6165
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  year     (Intercept) 0.005347 0.07312
##  Residual             4.344106 2.08425
## Number of obs: 3985, groups:  year, 15
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   3.6947     0.0384   96.22
```

```
m0.fe <- lm(partyid7 ~ factor(year), data=pid)
summary(m0.fe)
```

```
##
## Call:
## lm(formula = partyid7 ~ factor(year), data = pid)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8758 -1.8191 -0.5091  2.1615  3.6648
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.73234    0.12707  29.372   <2e-16 ***
## factor(year)1974 -0.02458    0.18969  -0.130   0.8969
## factor(year)1976  0.10496    0.18271   0.574   0.5657
## factor(year)1978 -0.08402    0.17155  -0.490   0.6243
## factor(year)1980  0.14347    0.21104   0.680   0.4966
## factor(year)1982 -0.39711    0.20206  -1.965   0.0494 *
## factor(year)1984  0.12480    0.17132   0.728   0.4664
## factor(year)1986 -0.22320    0.16580  -1.346   0.1783
## factor(year)1988  0.08677    0.17599   0.493   0.6220
## factor(year)1990 -0.11032    0.17051  -0.647   0.5177
## factor(year)1992 -0.08764    0.16909  -0.518   0.6043
## factor(year)1994  0.10617    0.17215   0.617   0.5375
## factor(year)1996  0.01031    0.17921   0.058   0.9541
## factor(year)1998 -0.24724    0.18609  -1.329   0.1841
## factor(year)2000  0.12480    0.27964   0.446   0.6554
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.084 on 3970 degrees of freedom
## Multiple R-squared:  0.004762,   Adjusted R-squared:  0.001253
## F-statistic: 1.357 on 14 and 3970 DF,  p-value: 0.1657
```

```
print(paste("Random Effects AIC =", AIC(m0)))
```

```
## [1] "Random Effects AIC = 17176.0681902987"
```

```
print(paste("Fixed Effects AIC =", AIC(m0.fe))) #In this case, really no difference whatsoever.
```

```
## [1] "Fixed Effects AIC = 17178.6709747081"
```

Now let's add a single explanatory variable, let's say `female`.

```
m1 <- lmer(partyid7 ~ 1 + female + (1 | year), data=pid)
summary(m1)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: partyid7 ~ 1 + female + (1 | year)
##    Data: pid
##
## REML criterion at convergence: 17164.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.3655 -0.8533 -0.2792  1.0445  1.6605
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  year     (Intercept) 0.004981 0.07058
##  Residual             4.335949 2.08229
## Number of obs: 3985, groups:  year, 15
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  3.79857    0.05181   73.32
## female      -0.19534    0.06614   -2.95
##
## Correlation of Fixed Effects:
```

```
##        (Intr)
## female -0.679
```

How can we pull fixed or random effects from the model?

```r
fixef(m1)
```

```
## (Intercept)      female 
##   3.7985729  -0.1953449
```

```r
coef(m1)
```

```
## $year
##      (Intercept)     female
## 1972    3.806596 -0.1953449
## 1974    3.801835 -0.1953449
## 1976    3.829893 -0.1953449
## 1978    3.785600 -0.1953449
## 1980    3.825009 -0.1953449
## 1982    3.737630 -0.1953449
## 1984    3.843466 -0.1953449
## 1986    3.745666 -0.1953449
## 1988    3.831618 -0.1953449
## 1990    3.776767 -0.1953449
## 1992    3.782408 -0.1953449
## 1994    3.837043 -0.1953449
## 1996    3.809538 -0.1953449
## 1998    3.754907 -0.1953449
## 2000    3.810618 -0.1953449
## 
## attr(,"class")
## [1] "coef.mer"
```

```r
ranef(m1)
```

```
## $year
##       (Intercept)
## 1972  0.008022891
## 1974  0.003262204
## 1976  0.031320058
## 1978 -0.012972817
## 1980  0.026435941
## 1982 -0.060943036
## 1984  0.044892753
## 1986 -0.052906764
## 1988  0.033044969
## 1990 -0.021805871
## 1992 -0.016164899
## 1994  0.038470137
## 1996  0.010965392
## 1998 -0.043666190
## 2000  0.012045233
```

So for a woman in 1996 we have $PID = 3.809 - 0.195X$. Also, the relationship between the `coef`, `fixef`, and `ranef`, is:

```r
identical( coef(m1)$year[1,1],
           as.numeric( fixef(m1)[1] + ranef(m1)$year[1,1] ) )
```

```
## [1] TRUE
```

```
#What does this mean?
```

We can also use canned functions from the `arm` package (`se.fixef` and `se.ranef`) to compute standard errors and confidence intervals. Here we'll look at the 95% confidence interval for the `female` variable.

```
fixef(m1)["female"] + qnorm(c(0.025, 0.975))*se.fixef(m1)["female"]
```

```
## [1] -0.32497270 -0.06571719
```

We can also look at the 95% confidence intervals for the random effects, like for the year 1972:

```
#Confidence interval for intercept
coef(m1)$year[1,1] + qnorm(c(0.025,0.975))*se.ranef(m1)$year[1]
```

```
## [1] 3.685690 3.927502
```

```
#What does this mean?



#Confidence interval for intercept deviation from common mean
ranef(m1)$year[1,] + qnorm(c(0.025, 0.975))*se.ranef(m1)$year[1]
```

```
## [1] -0.1128828  0.1289286
```

```
#What does this mean?
```

Next week we'll pick up with…the full model! And prediction!