Tutorial 13: Autocorrelation, Error Correction, and Model Simulations

Jan Vogler (jan.vogler@duke.edu)

December 2, 2016

Today's Agenda

1. Autocorrelation
2. Panel-corrected standard errors
3. Clustered standard errors
4. Model simulations
5. Using model simulations to estimate substantive effects
6. Learning R - what to do next?

1. Autocorrelation

Let us first load the LDC dataset.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/ps630_f16")
library(foreign)
LDC = read.dta("LDC_IO_replication.dta")
```

We use a regression that we have already seen in the past to illustrate the phenomenon of autocorrelation. This is our standard regression in which we analyze the impact of democratization on tariff levels.

```
lm_basic = lm(newtar ~ l1polity + l1gdp_pc + l1lnpop + l1ecris2 + l1bpc1 + l1avnewtar +
    factor(ctylabel) - 1, data = LDC)
# summary(lm_main)
```

There might be autocorrelation in our model because our units at time t and our units at time t-1 are likely to have similar values for the dependent and independent variables. Let us regress the residuals of our models on the residuals at time t-1:

```
res_t0 = lm_basic$resid
res_t1 = c(lm_basic$resid[2:length(lm_basic$resid)], NA)
res_data = as.data.frame(cbind(res_t1, res_t0))
head(res_data)
```

```
##         res_t1      res_t0
## 15 -7.42623616  6.99807104
## 16 -3.88875485 -7.42623616
## 17  4.07686936 -3.88875485
## 18  0.04671752  4.07686936
## 19 -2.67290721  0.04671752
## 24  5.54727523 -2.67290721
```

```r
lm_res = lm(res_t1 ~ res_t0, data = res_data)
summary(lm_res)
```

```
##
## Call:
## lm(formula = res_t1 ~ res_t0, data = res_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.813  -2.691  -0.068   2.451  31.918
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.006588   0.222448   -0.03    0.976
## res_t0       0.544863   0.031016   17.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.018 on 730 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.2971, Adjusted R-squared:  0.2962
## F-statistic: 308.6 on 1 and 730 DF,  p-value: < 2.2e-16
```

What we can see from this simple regression is that there is a high level of autocorrelation in our errors. Note that this code can easily be extended to include multiple lags of our residuals, which you might want to do to check for autocorrelation for several observations.

To address this problem, if we have enough datapoints available, we can create lags of our dependent variable (and other variables) and then include them in a new regression. The *DataCombine* package is designed for data management, especially time series and panel data. It allows us to easily generate lags of our dependent variable.

install.packages("DataCombine")

```r
library(DataCombine)

# summary(LDC)
```

We first define which variables we want to lag and we create a vector that includes prefixes for our lagged variables so that we can easily distinguish them from the non-lagged variables.

```r
# Choose the variables you want to lag
toLag = c("newtar","polityiv_update2")

# Define vector with the lag numbers
numberLag = c("lag1_","lag2_","lag3_")

# For loop to lag the data
for (i in 1:3){
  for (lagVar in toLag){
    LDC=slide(LDC, Var=lagVar, # Specify variable to lag
                  TimeVar="date", # Specificy time variable
                  GroupVar="ctylabel", # Unit variable
```

```
                  NewVar=paste0(numberLag[i],lagVar), # Name of new variable
                  slideBy = -i, # Lag by how many units, minus -> past
                  keepInvalid = FALSE, # Keep observations for which no lag can be created
                  reminder = TRUE) # Remind you to order the data by group variable
  }
}
```

```
##
## Lagging newtar by 1 time units.

##
## Lagging polityiv_update2 by 1 time units.

##
## Lagging newtar by 2 time units.

##
## Lagging polityiv_update2 by 2 time units.

##
## Lagging newtar by 3 time units.

##
## Lagging polityiv_update2 by 3 time units.
```

Let us now create a new model that contains three lags of the dependent variable.

```
lm_lag_dv = lm(newtar ~ lag1_newtar + lag2_newtar + lag3_newtar + l1polity +
    l1gdp_pc + l1lnpop + l1ecris2 + l1bpc1 + l1avnewtar + factor(ctylabel) -
    1, data = LDC)

# summary(lm_lag_dv)
```

How can we interpret the results of this model?

Also, there might be a time trend. Tariff levels might be moving up or down over time. If we can capture such a time trend with our model, it might reduce the autocorrelation in our errors.

```
lmTime = lm(newtar ~ lag1_newtar + lag2_newtar + lag3_newtar + l1polity + l1gdp_pc +
    l1lnpop + l1ecris2 + l1bpc1 + l1avnewtar + date + factor(ctylabel) - 1,
    data = LDC)

# summary(lmTime)
```

How would we interpret the results of our regression?

2. Panel-corrected standard errors

When we deal with variables that are correlated over time, i.e. variables that are "sticky" and exhibit little change from year to year, we have to account for this corrrelation somehow. In panel data (cross-sectional, time-series), a standard option is to use panel-corrected standard errors (PCSE).

Let us use a new dataset to illustrate panel-corrected standard errors. This dataset was downloaded from: http://people.stern.nyu.edu/wgreene/Econometrics/PanelDataSets.htm

According to the website it is from the dissertation of Y. Grunfeld (Univ. of Chicago, 1958).

The variables have the following meaning:

Firm = Firm ID, 1,...,10

Year = 1935,...,1954

I = Investment

F = Real Value of the Firm

C = Real Value of the Firm's Capital Stock

```r
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/ps630_f16/w13")
grunfeld = read.csv("grunfeld.csv")
```

Let us first load the "pcse" package: install.packages("pcse")

```r
library(pcse)
```

Now let us run a regression that includes panel-corrected standard errors.

We might think that the investment a firm makes is driven by the real value of the firm and the real value of its capital stock. We first have to run the regular regression.

```r
reg1 = lm(I ~ F + C, data = grunfeld)
summary(reg1)
```

```
##
## Call:
## lm(formula = I ~ F + C, data = grunfeld)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -291.68  -30.01    5.30   34.83  369.45
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -42.714369   9.511676  -4.491 1.21e-05 ***
## F             0.115562   0.005836  19.803  < 2e-16 ***
## C             0.230678   0.025476   9.055  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.41 on 197 degrees of freedom
## Multiple R-squared:  0.8124, Adjusted R-squared:  0.8105
## F-statistic: 426.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

As we can see, both are positive and highly significant. However, we have not considered the possibility that observations are correlated over time within the same units. That is why we need panel-data error correction.

Let us apply this.

```r
reg1pcse = pcse(reg1, groupN = grunfeld$FIRM, groupT = grunfeld$YEAR, pairwise = T)
```

Now let us obtain a new summary.

```r
summary(reg1pcse)
```

```
##
##  Results:
##
##                Estimate        PCSE   t value     Pr(>|t|)
## (Intercept) -42.7143694 6.780964847 -6.299158 1.913536e-09
## F             0.1155622 0.007212438 16.022621 1.538361e-37
## C             0.2306785 0.027886213  8.272134 1.943466e-14
##
##  ---------------------------------------------
##
## # Valid Obs = 200; # Missing Obs = 0; Degrees of Freedom = 197.
```

Does this differ from the regular regression? What are your conclusions?

By the way, here is how you obtain the corrected variance-covariance matrix. This might be useful for some applications.

```r
vcov_pcse = vcovPC(reg1, groupN = grunfeld$FIRM, groupT = grunfeld$YEAR, pairwise = TRUE)
```

Compare this to the regular one.

```r
vcov_reg = vcov(reg1)

vcov_pcse
```

```
##               X.Intercept.             F             C
## X.Intercept.   45.98148426 -1.507501e-02 -0.1092682575
## F              -0.01507501  5.201926e-05 -0.0001050313
## C              -0.10926826 -1.050313e-04  0.0007776409
```

```r
vcov_reg
```

```
##             (Intercept)             F             C
## (Intercept) 90.47198093 -1.678301e-02 -1.005495e-01
## F           -0.01678301  3.405551e-05 -7.265559e-05
## C           -0.10054949 -7.265559e-05  6.490165e-04
```

3. Clustered standard errors

We use clustered standard errors to correct for correlation among all observations of a specific subgroup of the data.

For example, we might want to use clustered standard errors, when we run regressions on the vote1 dataset that we have used in previous problem sets.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/ps630_f16/w13")

library(foreign)
vote1 = read.dta("VOTE1.dta")
summary(vote1)
```

```
##     state              district         democA           voteA
##  Length:173         Min.   : 1.000   Min.   :0.0000   Min.   :16.0
##  Class :character   1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:36.0
##  Mode  :character   Median : 6.000   Median :1.0000   Median :50.0
##                     Mean   : 8.838   Mean   :0.5549   Mean   :50.5
##                     3rd Qu.:11.000   3rd Qu.:1.0000   3rd Qu.:65.0
##                     Max.   :42.000   Max.   :1.0000   Max.   :84.0
##     expendA            expendB          prtystrA        lexpendA
##  Min.   :   0.302   Min.   :   0.93   Min.   :22.00   Min.   :-1.197
##  1st Qu.:  81.634   1st Qu.:  60.05   1st Qu.:44.00   1st Qu.: 4.402
##  Median : 242.782   Median : 221.53   Median :50.00   Median : 5.492
##  Mean   : 310.611   Mean   : 305.09   Mean   :49.76   Mean   : 5.026
##  3rd Qu.: 457.410   3rd Qu.: 450.72   3rd Qu.:56.00   3rd Qu.: 6.126
##  Max.   :1470.674   Max.   :1548.19   Max.   :71.00   Max.   : 7.293
##     lexpendB            shareA
##  Min.   :-0.07257   Min.   : 0.09464
##  1st Qu.: 4.09524   1st Qu.:18.86800
##  Median : 5.40056   Median :50.84990
##  Mean   : 4.94437   Mean   :51.07654
##  3rd Qu.: 6.11084   3rd Qu.:84.25510
##  Max.   : 7.34484   Max.   :99.49500
```

```
lm_vote = lm(voteA ~ expendA + expendB + prtystrA, data = vote1)
summary(lm_vote)
```

```
##
## Call:
## lm(formula = voteA ~ expendA + expendB + prtystrA, data = vote1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.661  -8.385   0.362   8.536  30.814
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.267190   4.416784   7.532 2.87e-12 ***
## expendA      0.034924   0.003369  10.365  < 2e-16 ***
## expendB     -0.034924   0.003001 -11.636  < 2e-16 ***
## prtystrA     0.342514   0.087952   3.894 0.000142 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.12 on 169 degrees of freedom
## Multiple R-squared:  0.5687, Adjusted R-squared:  0.561
## F-statistic: 74.27 on 3 and 169 DF,  p-value: < 2.2e-16
```

In order to use clustered standard errors, we first have to install multiple packages.

install.packages("plm") install.packages("lmtest") install.packages("multiwayvcov")

```r
library(plm)
```

```
## Warning: package 'plm' was built under R version 3.3.2
```

```
## Loading required package: Formula
```

```r
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 3.3.2
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(multiwayvcov)
```

```
## Warning: package 'multiwayvcov' was built under R version 3.3.2
```

Now we have to obtain the corrected variance-covariance matrix.

```r
lm_vote.vcovCL = cluster.vcov(lm_vote, vote1$state)

lm_vote_CSE = coeftest(lm_vote, lm_vote.vcovCL)

lm_vote_CSE
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 33.2671901  5.7963348   5.7393 4.306e-08 ***
## expendA      0.0349245  0.0044572   7.8355 4.957e-13 ***
## expendB     -0.0349236  0.0031879 -10.9549 < 2.2e-16 ***
## prtystrA     0.3425140  0.1055593   3.2448  0.001417 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Have our results changed? How would you interpret these new findings?

4. Model simulations

Credit to Professor Chris Johnston. The code for the simulation approach introduced here is from his class.

We will look at new data that we have not used previously: data on attitudes toward the United States Supreme Court. This is a topic that students of political behavior and identities might be interested in.

```
setwd("C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/ps630_f16/w13")
courtdata <- read.table("courtdata.txt", header = TRUE)
# summary(courtdata)
```

What is the meaning of the variables in the data set?

Independent Variables

college = dummy for college education (1 = college educated)

ideo = 5-point ideological self-identification, ranging from "very liberal" (1) to "very conservative" (5)

soph = 10-item political knowledge scale (0 = low, 10 = high)

black & hispanic = dummies for respective categories (1 = black, 1 = hispanic)

income = 15-point household income category scale (1 = low, 15 = high)

ruling = dummy indicating respondent correctly identified the Affordable Care Act ruling (1)

male = dummy indicating male gender (1 = male)

Dependent Variable

We are interested in attitudes of US citizens toward the Supreme Court. For this purpose we look at the variable **sclaw** (standing for "Supreme Court Law"). The variable was based on the following statement:

"The Supreme Court should be allowed to throw out any law it deems unconstitutional"

The variable ranges from "strongly agree" to "strongly disagree".

Let us estimate a linear model.

```
model1 <- lm(sclaw ~ ruling + ideo + soph + age + male + black + hisp + college +
    income, data = courtdata)
summary(model1)
```

```
##
## Call:
## lm(formula = sclaw ~ ruling + ideo + soph + age + male + black +
##     hisp + college + income, data = courtdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9065 -1.1406 -0.3534  0.8163  3.2797
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  2.919930   0.240242  12.154  < 2e-16 ***
## ruling      -0.133350   0.135592  -0.983  0.32567
## ideo        -0.135789   0.045475  -2.986  0.00291 **
## soph        -0.036333   0.020474  -1.775  0.07633 .
## age          0.004780   0.003167   1.509  0.13161
## male        -0.165054   0.100291  -1.646  0.10019
## black        0.071503   0.153606   0.465  0.64170
## hisp        -0.123167   0.170601  -0.722  0.47052
## college     -0.039149   0.105591  -0.371  0.71091
## income       0.009154   0.012308   0.744  0.45721
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.34 on 835 degrees of freedom
## Multiple R-squared:  0.02962,    Adjusted R-squared:  0.01916
## F-statistic: 2.832 on 9 and 835 DF,  p-value: 0.002746
```

When we estimate a linear model, all our coefficients are assumed to be normally distributed random variables with a mean and a variance that depends on the data. We can make use of this fact by simulating different versions of the model in which the coefficients are normally distributed around the original model.

For the simulations we need the package "arm". Please make sure to install it via the following command:

install.packages("arm")

```r
library(arm)
```

```
## Loading required package: MASS

## Loading required package: Matrix

## Loading required package: lme4

##
## arm (Version 1.9-1, built: 2016-8-21)

## Working directory is C:/Users/Jan/OneDrive/Documents/GitHub/ps630_lab/ps630_f16/W13
```

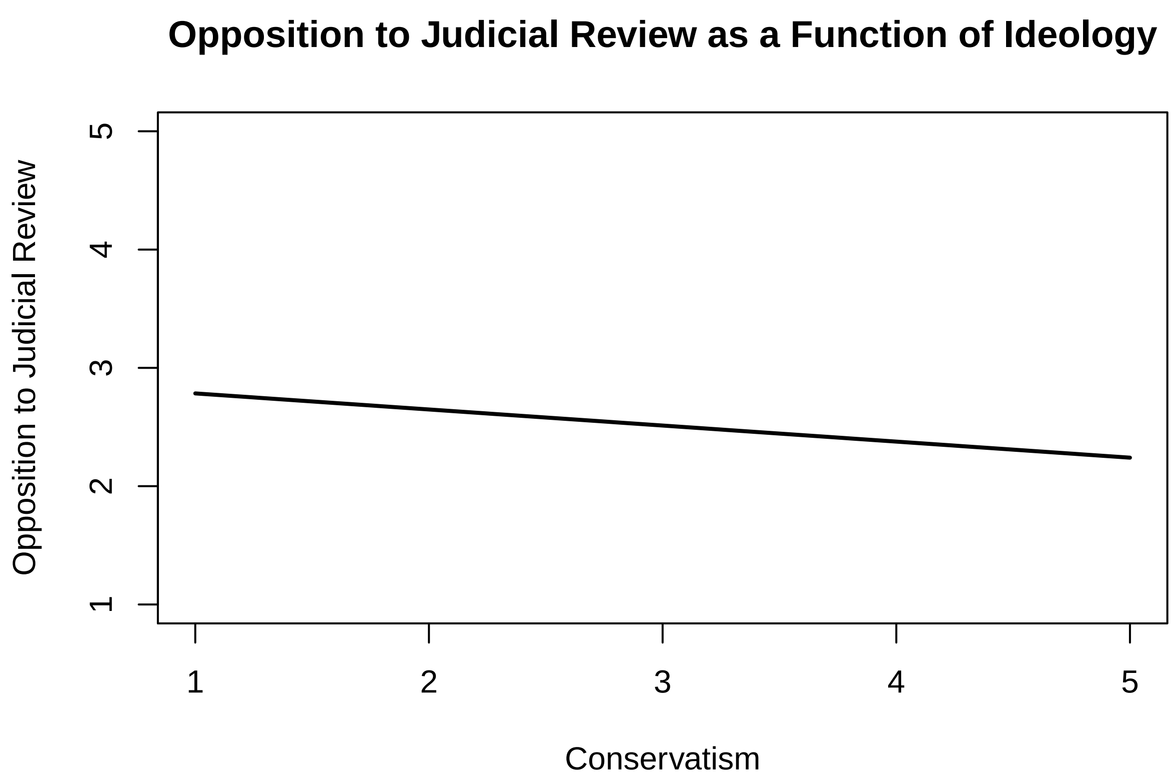
```r
model1.sims <- sim(model1, n.sims = 1000)
```

How did we create these simulations? How could that be useful to us?

5. Using model simulations to estimate substantive effects

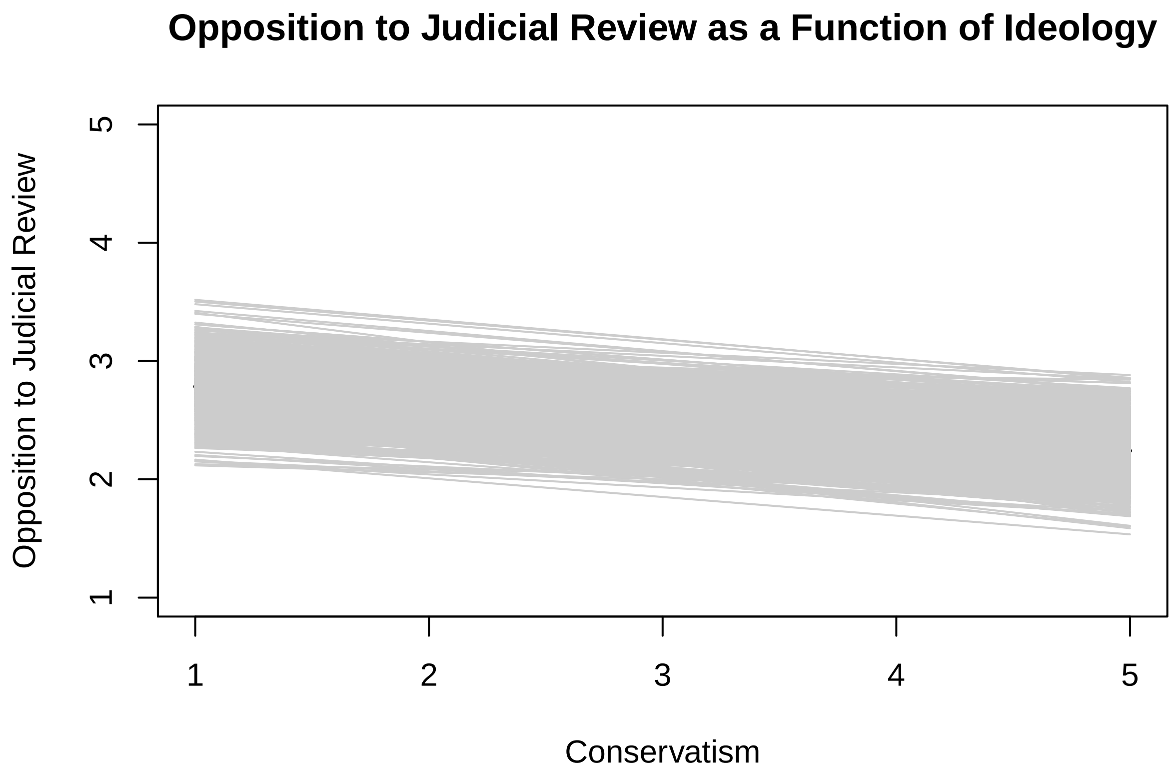
The following plot is generated by our knowledge about the regression. We access the first coefficient (the intercept) and the third coefficient (ideology). We let ideology vary from 1 to 5. If we plug in the right formula, then we will get the predicted values when all other variables are at the value 0. (This is similar to our predictions from previous tutorials, where we held all variables at their mean.)

```
curve(coef(model1)[1] + coef(model1)[3] * x, from = 1, to = 5, ylim = c(1, 5),
    xlab = "Conservatism", ylab = "Opposition to Judicial Review", main = "Opposition to Judicial Revie
    lwd = 2)
```



Now let's look instead at the lines that are the result of the 1000 simulations that we created above.

```
curve(coef(model1)[1] + coef(model1)[3] * x, from = 1, to = 5, ylim = c(1, 5),
    xlab = "Conservatism", ylab = "Opposition to Judicial Review", main = "Opposition to Judicial Revie
    lwd = 2)
for (i in 1:1000) {
    curve(coef(model1.sims)[i, 1] + coef(model1.sims)[i, 3] * x, add = TRUE,
        col = "gray80")
}
```

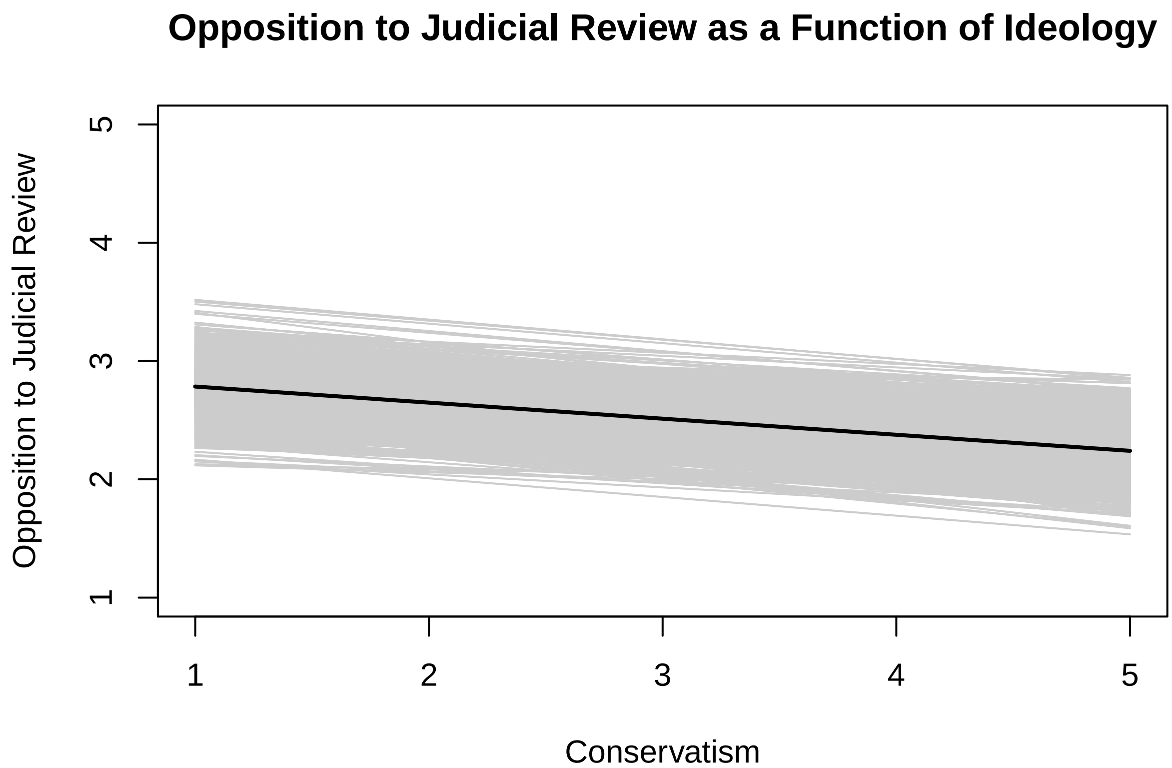


What we can see in this plot is that the simulated coefficients are very similar to the one that we already had. In fact, the simulation predictions are approximately normally distributed around our original regression line.

Let's run the original command one more time, so we get our regression line back on top of everything else.

```
curve(coef(model1)[1] + coef(model1)[3] * x, from = 1, to = 5, ylim = c(1, 5),
    xlab = "Conservatism", ylab = "Opposition to Judicial Review", main = "Opposition to Judicial Revie
    lwd = 2)
for (i in 1:1000) {
    curve(coef(model1.sims)[i, 1] + coef(model1.sims)[i, 3] * x, add = TRUE,
        col = "gray80")
}

curve(coef(model1)[1] + coef(model1)[3] * x, col = "black", lwd = 2, add = TRUE)
```



Let us now utilize our model simulations. They can help us to make an average predictive comparison. We use our model simulations in combination with draws from the data to create a so-called "average predictive comparison". These help us to evaluate the substantive impact that a variable has in comparison with other variables. Our results will include confidence intervals.

In order to do this, we first create an array that we fill with data. We have 1000 simulations of the model and X observations or data points. We will apply each of those 1000 simulations to all our data points to estimate the average predicted effect.

```
d.ideo <- array(NA, c(1000, length(courtdata$sclaw)))
m.ideo <- array(NA, 1000)
```

Now we run a for loop. In this for loop we go through each of the 1000 simulations of the model that we have generated based on the variance-covariance matrix. An in each of the 1000 iterations, we also go through the data in its entirety. In each of these iterations, we subtract the effect of holding our ideology at the minimum from holding it at the maximum. This will give us an idea of what the effect is across many possible models and all of the empirical data that is available to us - a so-called "average prediction" of the effect.

Please note that the following code applies to all kinds of models, including such that have non-linear link functions. For linear functions, such as linear regression, we can reduce and simplify this code. The full code is provided to ensure that you can also use it with respect to other types of models, especially logit and probit models, where variables generally do not have effects that are directly proportional.

```
# Remember our model was: sclaw ~ ruling + ideo + soph + age + male + black
# + hisp + college + income
```

```
summary(courtdata$ideo)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   3.000   3.128   4.000   5.000
```

```
for (i in 1:1000) {
    d.ideo[i, ] <- (coef(model1.sims)[i, 1] + coef(model1.sims)[i, 2] * courtdata$ruling +
        coef(model1.sims)[i, 3] * 5 + coef(model1.sims)[i, 4] * courtdata$soph +
        coef(model1.sims)[i, 5] * courtdata$age + coef(model1.sims)[i, 6] *
        courtdata$male + coef(model1.sims)[i, 7] * courtdata$black + coef(model1.sims)[i,
        8] * courtdata$hisp + coef(model1.sims)[i, 9] * courtdata$college +
        coef(model1.sims)[i, 10] * courtdata$income) - (coef(model1.sims)[i,
        1] + coef(model1.sims)[i, 2] * courtdata$ruling + coef(model1.sims)[i,
        3] * 0 + coef(model1.sims)[i, 4] * courtdata$soph + coef(model1.sims)[i,
        5] * courtdata$age + coef(model1.sims)[i, 6] * courtdata$male + coef(model1.sims)[i,
        7] * courtdata$black + coef(model1.sims)[i, 8] * courtdata$hisp + coef(model1.sims)[i,
        9] * courtdata$college + coef(model1.sims)[i, 10] * courtdata$income)
    m.ideo[i] <- mean(d.ideo[i, ])
}

mean(m.ideo)
```

```
## [1] -0.6791549
```

```
sd(m.ideo)
```

```
## [1] 0.2226786
```

```
quantile(m.ideo, probs = c(0.025, 0.16, 0.84, 0.975))
```

```
##       2.5%        16%        84%      97.5%
## -1.1208675 -0.9029604 -0.4617737 -0.2549308
```

Let's compare this to the effect that sophistication has. We need to also estimate the average predictive effects for sophistication. Please note that this time we let ideology vary with the data. What we keep constant here is the sophistication variable at its minimum and maximum to assess the average effect that it has on attitudes toward the Supreme Court.

```
summary(courtdata$soph)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.000   7.000   6.724   9.000  10.000
```

```
d.soph <- array(NA, c(1000, length(courtdata$sclaw)))
m.soph <- array(NA, 1000)

for (i in 1:1000) {
    d.soph[i, ] <- ((coef(model1.sims)[i, 1] + coef(model1.sims)[i, 2] * courtdata$ruling +
        coef(model1.sims)[i, 3] * courtdata$ideo + coef(model1.sims)[i, 4] *
```

```
        10 + coef(model1.sims)[i, 5] * courtdata$age + coef(model1.sims)[i,
        6] * courtdata$male + coef(model1.sims)[i, 7] * courtdata$black + coef(model1.sims)[i,
        8] * courtdata$hisp + coef(model1.sims)[i, 9] * courtdata$college +
        coef(model1.sims)[i, 10] * courtdata$income) - (coef(model1.sims)[i,
        1] + coef(model1.sims)[i, 2] * courtdata$ruling + coef(model1.sims)[i,
        3] * courtdata$ideo + coef(model1.sims)[i, 4] * 0 + coef(model1.sims)[i,
        5] * courtdata$age + coef(model1.sims)[i, 6] * courtdata$male + coef(model1.sims)[i,
        7] * courtdata$black + coef(model1.sims)[i, 8] * courtdata$hisp + coef(model1.sims)[i,
        9] * courtdata$college + coef(model1.sims)[i, 10] * courtdata$income))
    m.soph[i] <- mean(d.soph[i, ])
}

mean(m.soph)
```

```
## [1] -0.3504821
```

```
sd(m.soph)
```

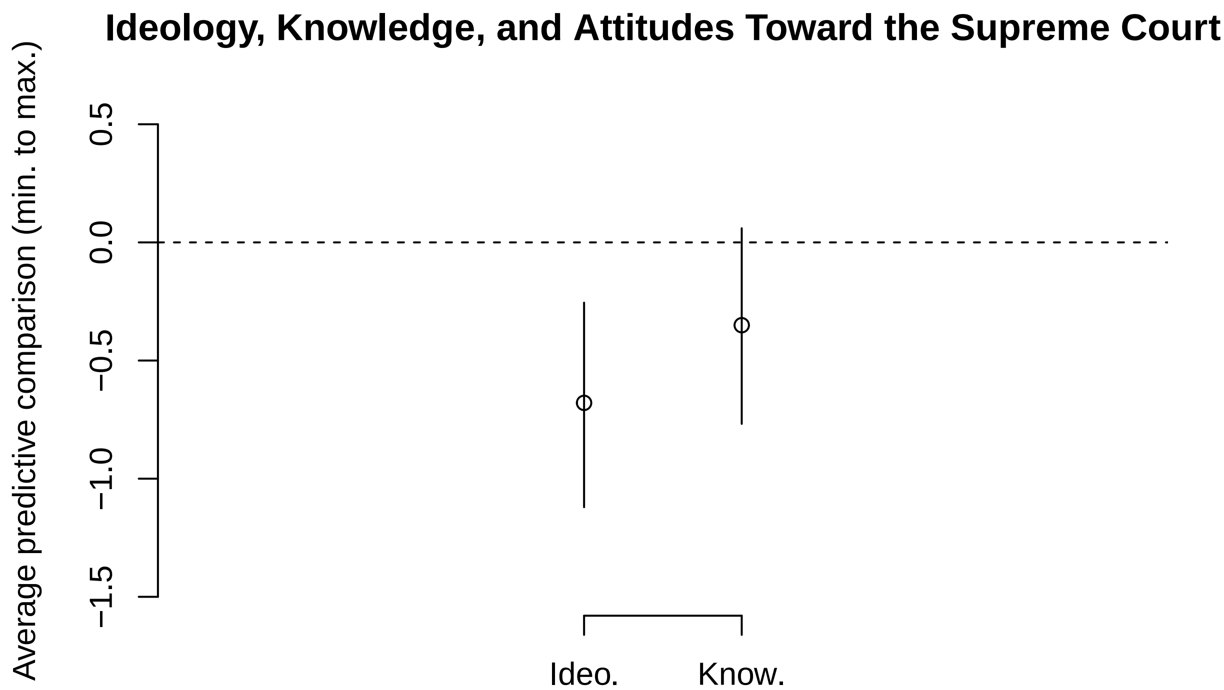
```
## [1] 0.2088545
```

```
quantile(m.soph, probs = c(0.025, 0.16, 0.84, 0.975))
```

```
##        2.5%         16%         84%       97.5%
## -0.76811165 -0.55083765 -0.15025953  0.05990856
```

Let us plot these two in comparison.

```
plot(1:2, c(mean(m.ideo), mean(m.soph)), type = "p", ylim = c(-1.5, 0.5), xlab = "",
    main = "Ideology, Knowledge, and Attitudes Toward the Supreme Court", ylab = "Average predictive co
    asp = 1.5, axes = FALSE)
axis(1, at = c(1, 2), labels = c("Ideo.", "Know."))
axis(2, at = c(-1.5, -1, -0.5, 0, 0.5))
abline(h = 0, lty = 2)
segments(1, quantile(m.ideo, probs = c(0.025)), 1, quantile(m.ideo, probs = c(0.975)))
segments(2, quantile(m.soph, probs = c(0.025)), 2, quantile(m.soph, probs = c(0.975)))
```



Which variable has the greater substantive effect? What can we say about the average influence of the knowledge variable?

6. Learning R - what to do next?

Here are some pieces of advice on how to proceed from here.

First, when you have some free time during the winter break, use it to review what you have learned in this class. You will need it in the future.

The following classes are most useful in terms of pushing your R skills further:

1. PolSci 733 - *Maximum Likelihood Methods* (2nd semester)
2. Stats 523 - *Statistical Programming* (3rd semester/5th semester)
3. Stats 601 - *Bayesian and Modern Statistics* (4th semester/6th semester)

Note: R is best learned gradually. Don't take everything at once. Give yourself the opportunity to learn more in the future.

Also, Stats 250 has an R lab, too. This R lab focuses on introducing R for mathematical purposes in the context of statistical theory (and it does not assume a background in R).

The following books will help you to master R:

1. Fox & Weisberg (2010) *An R Companion to Applied Regression* (2nd Edition)
2. Gelman & Hill (2006) *Data Analysis Using Regression and Multilevel/Hierarchical Models*

3. Chang (2013) *R Graphics Cookbook*
4. Kabacoff (2011) *R in Action*
5. Maindonald & Braun (2010) *Data analysis and graphics using R*
6. Matloff (2011) *The Art of R Programming - a Tour of Statistical Software Design*

In the future, you will often find yourself in the situation that the tutorial has not covered a problem you run into. Then you need to be able to find a solution yourself. Taking advanced classes in R will also help you to *develop* a solution yourself, i.e. to write your own script in R to deal with the problem.

R is a continuous learning experience.

Remember

From our first session.

```r
fun = c("R", "is", "fun")
paste(fun, collapse = " ")
```

```
## [1] "R is fun"
```