Tutorial 11: Interaction terms

Johannes Karreth

RPOS 517, Day 11

This tutorial shows you:

- how to specify regression models with interaction terms
- (briefly) how to interpret interaction terms (refer to *AR* ch. 7 and Brambor et al. for a more detailed treatment)
- how to graph marginal/conditional effects from regression estimates

Note on copying & pasting code from the PDF version of this tutorial: Please note that you may run into trouble if you copy & paste code from the PDF version of this tutorial into your R script. When the PDF is created, some characters (for instance, quotation marks or indentations) are converted into non-text characters that R won't recognize. To use code from this tutorial, please type it yourself into your R script or you may copy & paste code from the *source file* for this tutorial which is posted on my website.

Note on R functions discussed in this tutorial: I don't discuss many functions in detail here and therefore I encourage you to look up the help files for these functions or search the web for them before you use them. This will help you understand the functions better. Each of these functions is well-documented either in its help file (which you can access in R by typing `?ifelse`, for instance) or on the web. The *Companion to Applied Regression* (see our syllabus) also provides many detailed explanations.

As always, please note that this tutorial only accompanies the other materials for Day 11 and that you are expected to have worked through the reading for that day before tackling this tutorial.

Note: You must have read Brambor, Clark & Golder (2006) before working on this tutorial.

Interaction terms in regression

Interaction terms enter the regression equation as the product of two *constitutive* terms, x_1 and x_2. For this *product term* x_1x_3, the regression equation adds a separate coefficient β_3.

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

Example 1: one binary, one continuous term

In thinking about interaction terms, it helps to first simplify by working through the prediction of the regression equation for different values of two predictors, x_1 and x_2. We can imagine a continuous outcome y, e.g. the income of Hollywood actors, that we predict with two variables. The first, x_1, is a binary variable such as female gender; it takes on values of 0 (males) and 1 (females) only. The second, x_2, is a continuous variable, ranging from -5 to $+5$, such as a (centered and standardized) measure of age. In this case, I'm using the term "effect" loosely and non-causally. An interaction term expresses the idea that the effect of one variable depends on the value of the other variable. With these variables, this suggests that effect of age on actors' income is different for male actors than for female actors.

- β_1 is the effect of x_1 on y when x_2 is 0:

  - $y = \alpha + \beta_1 x_1 + \beta_2 \times 0 + \beta_3 x_1 \times 0$
  - $y = \alpha + \beta_1 x_1 + 0 + 0$

- β_2 is the effect of x_2 on y when x_1 is 0:

  - $y = \alpha + \beta_1 \times 0 + \beta_2 x_2 + \beta_3 \times 0 \times x_2$
  - $y = \alpha + \beta_2 x_2 + 0$

- When both x_1 and x_2 are **not** 0, β_3 becomes important, and the effect of x_1 now varies with the value of x_2. We can plug in 1 for x_1 and simplify the equation as follows:

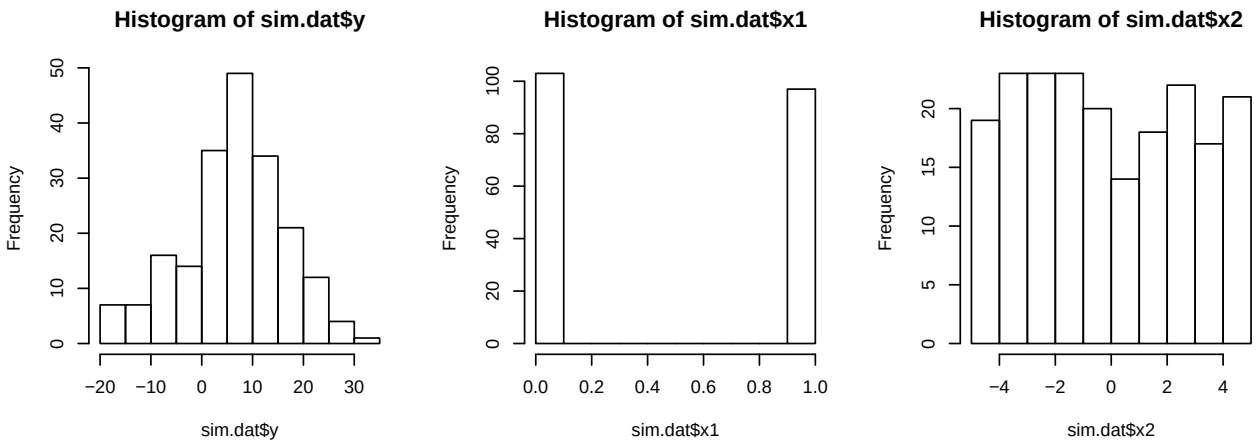
  - $y = \alpha + \beta_1 \times 1 + \beta_2 x_2 + \beta_3 \times 1 \times x_2$
  - $y = \alpha + \beta_1 + \beta_2 x_2 + \beta_3 \times x_2$
  - $y = (\alpha + \beta_1) + (\beta_2 + \beta_3) \times x_2$

With simulated data, this can be illustrated easily. I begin by simulating a dataset with 200 observations, two predictors x_1 (binary: male/female) and x_2 (continuous: standardized and centered age), and create y (continuous: income) as the linear combination of x_1, x_2, and an interaction term of the two predictors.

```
set.seed(123)
n.sample <- 200
x1 <- rbinom(n.sample, size = 1, prob = 0.5)
x2 <- runif(n.sample, -5, 5)
a <- 5
b1 <- 3
b2 <- 4
b3 <- -3
e <- rnorm(n.sample, 0, 5)
y <- a + b1 * x1 + b2 * x2 + b3 * x1 * x2 + e
sim.dat <- data.frame(y, x1, x2)
```

Before advancing, this is what the simulated data look like:

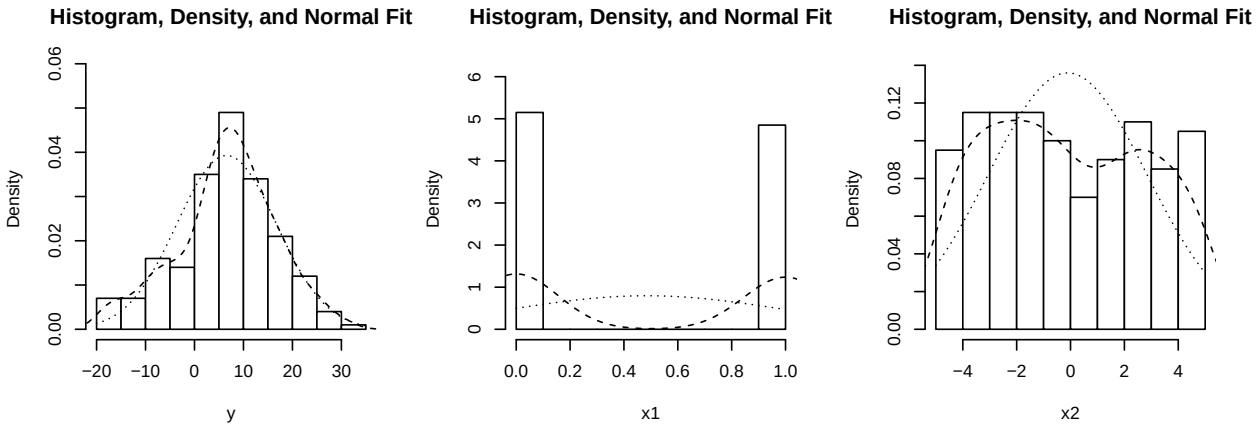
```
par(mfrow = c(1, 3))
hist(sim.dat$y)
hist(sim.dat$x1)
hist(sim.dat$x2)
```



```r
par(mfrow = c(1, 1))
```

For convenience, you could also use the `multi.hist()` function from the "psych" package. It automatically adds a density curve (dashed line) and a normal density plot (dotted line).

```r
# install.packages(psych)
library(psych)
multi.hist(sim.dat, nrow = 1)
```



Fitting a model using what we know about the data-generating process gives us:

```r
mod.sim <- lm(y ~ x1 * x2, dat = sim.dat)
summary(mod.sim)
```

```
##
## Call:
## lm(formula = y ~ x1 * x2, data = sim.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.1974  -3.4874  -0.0738   3.4837  13.2178
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.7891     0.4924   9.726  < 2e-16 ***
## x1            3.8716     0.7081   5.467 1.38e-07 ***
## x2            3.9554     0.1663  23.790  < 2e-16 ***
## x1:x2        -2.9435     0.2421 -12.160  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.997 on 196 degrees of freedom
## Multiple R-squared:  0.7615, Adjusted R-squared:  0.7579
## F-statistic: 208.6 on 3 and 196 DF,  p-value: < 2.2e-16
```

First, I plot the relationship between the continuous predictor x_2 and y for all observations where $x_1 = 0$. The regression line is defined by an intercept α and a slope β_2, as we calculated above. I can extract these from the `lm` object above using the `coef()` function. I use the index in square brackets to extract the coefficient I need.

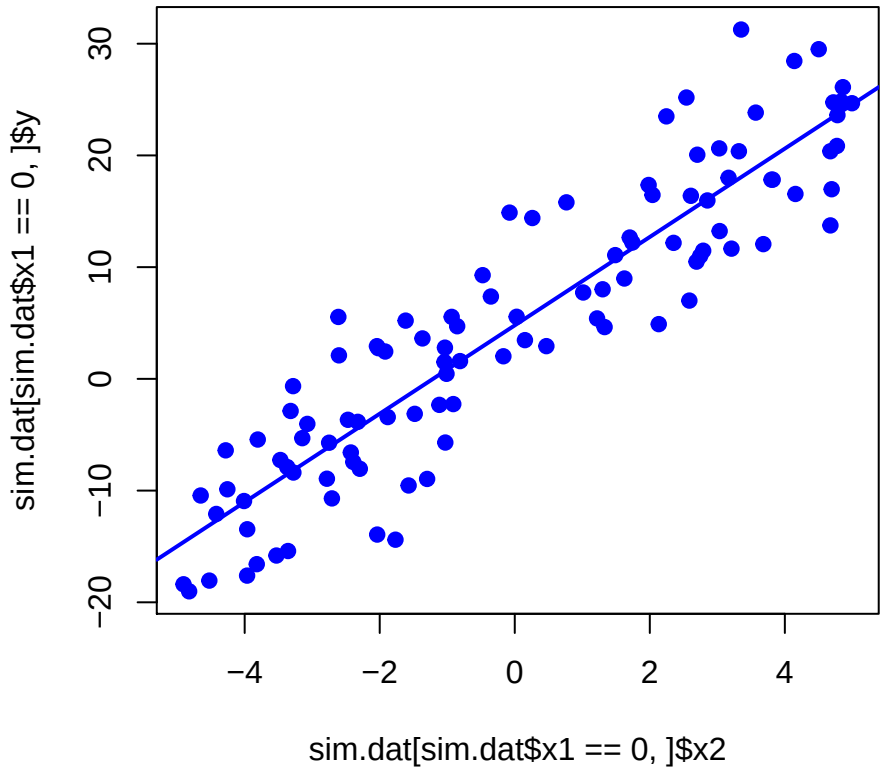
```
coef(mod.sim)
```

```
## (Intercept)          x1          x2       x1:x2
##    4.789084    3.871614    3.955383   -2.943516
```

```
coef(mod.sim)[1]
```

```
## (Intercept)
##    4.789084
```

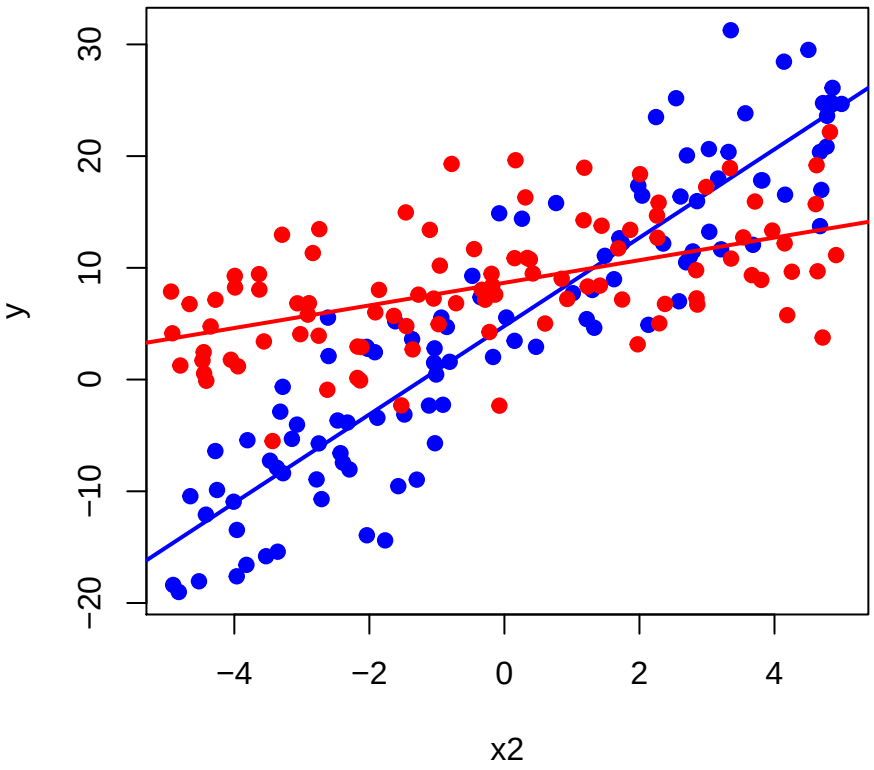
```
plot(x = sim.dat[sim.dat$x1 == 0, ]$x2, y = sim.dat[sim.dat$x1 == 0, ]$y,
     col = "blue", pch = 19)
abline(a = coef(mod.sim)[1], b = coef(mod.sim)[3], col = "blue", pch = 19, lwd = 2)
```



1. For all cases where $x_1 = 0$, what is the relationship between x_2 and y?

Next, we can add the data points where $x_1 = 1$, and add the separate regression line for these points:

```
plot(x = sim.dat[sim.dat$x1 == 0, ]$x2, y = sim.dat[sim.dat$x1 == 0, ]$y,
     pch = 19, xlab = "x2", ylab = "y", col = "blue")
abline(a = coef(mod.sim)[1], b = coef(mod.sim)[3], col = "blue", lwd = 2)
points(x = sim.dat[sim.dat$x1 == 1, ]$x2, y = sim.dat[sim.dat$x1 == 1, ]$y,
       col = "red", pch = 19)
abline(a = coef(mod.sim)[1] + coef(mod.sim)[2], b = coef(mod.sim)[3] + coef(mod.sim)[4],
       col = "red", lwd = 2)
```



2. For all cases where $x_1 = 1$, what is the relationship between x_2 and y?

From this first exercise, you should take home a few things:

Specifying an interaction

- Although the tutorial does not discuss this, interaction terms should be included based on theory.
- In R, you specify an interaction term by putting an asterisk between the two constitutive terms: `x1 * x2`

You always need to include the constitutive terms in your model.

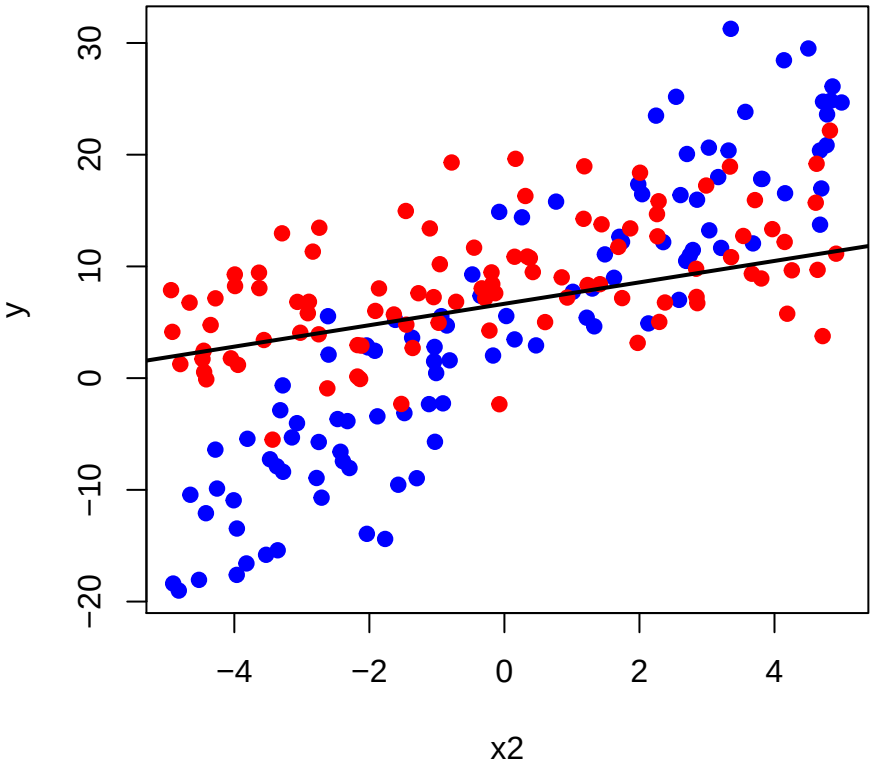
See Brambor et al. (2006) for more details. What would it imply if your model only included $\beta_3 x_1 x_2$? Omitting a coefficient is equivalen to setting $\beta_1 = 0$ and $\beta_2 = 0$. See for yourself:

```
mod.sim2 <- lm(y ~ x1:x2, data = sim.dat)
summary(mod.sim2)
```

```
##
## Call:
## lm(formula = y ~ x1:x2, data = sim.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.6719  -4.5727   0.6632   5.8826  24.6120
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.6555     0.7078   9.404  < 2e-16 ***
## x1:x2         0.9588     0.3513   2.729  0.00692 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.995 on 198 degrees of freedom
## Multiple R-squared:  0.03625,    Adjusted R-squared:  0.03138
## F-statistic: 7.448 on 1 and 198 DF,  p-value: 0.006924
```

```
plot(x = sim.dat[sim.dat$x1 == 0, ]$x2, y = sim.dat[sim.dat$x1 == 0, ]$y,
     pch = 19, xlab = "x2", ylab = "y", col = "blue")
points(x = sim.dat[sim.dat$x1 == 1, ]$x2, y = sim.dat[sim.dat$x1 == 1, ]$y,
       col = "red", pch = 19)
abline(a = coef(mod.sim2)[1], b = coef(mod.sim2)[2], lwd = 2)
```



Note: R will automatically include the constitutive terms when you use the asterisk as above.

- `y ~ x1 + x2 + x1 * x2` is equivalent to `y ~ x1 * x2`

Example 2: Two continuous variables

The first example illustrates the general logic of interaction terms. It carries over directly into the case of an interaction term between two continuous variables. Let's now simulate another (fake) dataset, with a continuous outcome y being the income of professional baseball players. We can assume that x_1 is a continuous variable and distributed normally with a mean of 2 and a standard deviation of 5 (e.g., a standardized measure of injury risk of each player). As previously, x_2 is a continuous variable (such as standardized &

centered age), ranging from -5 to $+5$. We might be thinking of a conditional effect, where older athletes make more — but only if they are rated as low injury risk. Older athletes with higher injury risk might be making less money. With this configuration,

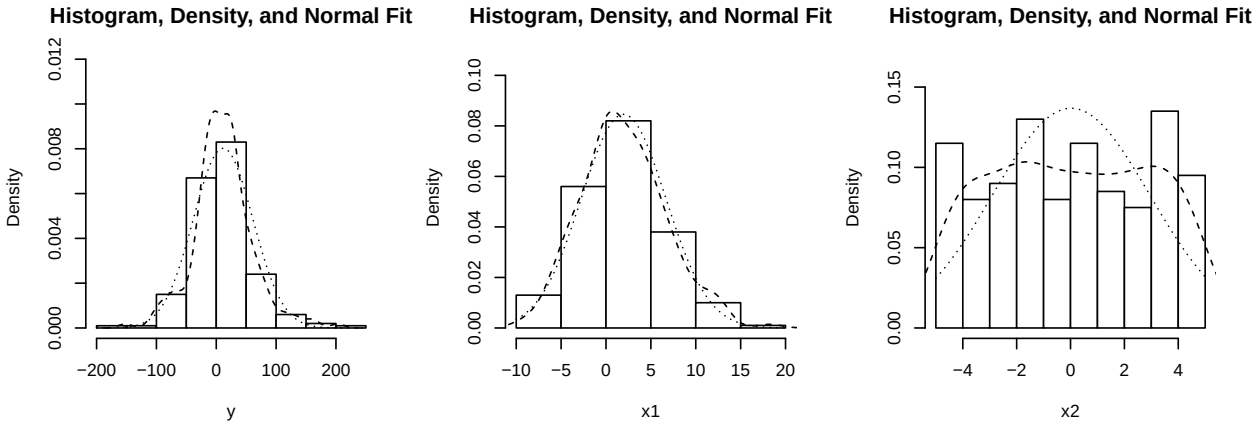
- β_1 is the effect of x_1 on y when x_2 is 0:
  - $y = \alpha + \beta_1 x_1 + \beta_2 \times 0 + \beta_3 x_1 \times 0$
  - $y = \alpha + \beta_1 x_1 + 0 + 0$
- β_2 is the effect of x_2 on y when x_1 is 0:
  - $y = \alpha + \beta_1 \times 0 + \beta_2 x_2 + \beta_3 \times 0 \times x_2$
  - $y = \alpha + \beta_2 x_2 + 0$
- When both x_1 and x_2 are **not** 0, β_3 becomes important, and the effect of x_1 now varies with the value of x_2. We can again plug in 1 for x_1 and simplify the equation as follows:
  - $y = \alpha + \beta_1 \times 1 + \beta_2 x_2 + \beta_3 \times 1 \times x_2$
  - $y = \alpha + \beta_1 + \beta_2 x_2 + \beta_3 \times x_2$
  - $y = (\alpha + \beta_1) + (\beta_2 + \beta_3) \times x_2$
- But x_1 takes on many other values than just 1. The more general equation for y is then (cf. Brambor et al. p. 11)
  - $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$
  - In this case, we cannot define just "two" equations (for $x_1 = 1$ and $x_1 = 0$). Instead, the effect of x_1 varies with x_2, and vice versa.
  - To express this, we use the term "marginal effects" or "conditional effects": the effect of x_1 conditional on the value of x_2.
  - This marginal effect for x_1 is a slope, and this slope is the sum of the coefficient for x_1 and the product of β_3 and the value of x_2: $\beta_1 + \beta_3 \times x_2$.
  - Similarly, the marginal effect for x_2 is $\beta_2 + \beta_3 \times x_1$.

I use the same simulation setup above, but make x_1 a continous variable, normally distributed with a mean of 2 and a standard deviation of 5.

```
set.seed(123)
n.sample <- 200
x1 <- rnorm(n.sample, mean = 2, sd = 5)
x2 <- runif(n.sample, -5, 5)
a <- 5
b1 <- 3
b2 <- 4
b3 <- -3
e <- rnorm(n.sample, 0, 20)
y <- a + b1 * x1 + b2 * x2 + b3 * x1 * x2 + e
sim.dat2 <- data.frame(y, x1, x2)
```

Before advancing, this is what the simulated data look like:

```
multi.hist(sim.dat2, nrow = 1)
```



Fitting a model using what we know about the data-generating process gives us:

```
mod.sim3 <- lm(y ~ x1 * x2, dat = sim.dat2)
summary(mod.sim3)
```

```
##
## Call:
## lm(formula = y ~ x1 * x2, data = sim.dat2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.539 -12.129   0.986  12.424  51.132
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.4950     1.5581   4.168 4.60e-05 ***
## x1            2.5923     0.3059   8.475 5.59e-15 ***
## x2            4.1026     0.5325   7.704 6.41e-13 ***
## x1:x2        -3.0383     0.1017 -29.883  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.34 on 196 degrees of freedom
## Multiple R-squared:  0.8349, Adjusted R-squared:  0.8323
## F-statistic: 330.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

We can now calculate the effect of x_1 on y at different levels of x_2, say, across the range of x_2 with increments of 1. In R, I create this sequence using the `seq()` function.

```
x2.sim <- seq(from = -5, to = 5, by = 1)
x2.sim
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
eff.x1 <- coef(mod.sim3)[2] + coef(mod.sim3)[4] * x2.sim
```

I can now list the effect of x_1 at different levels of x_2:

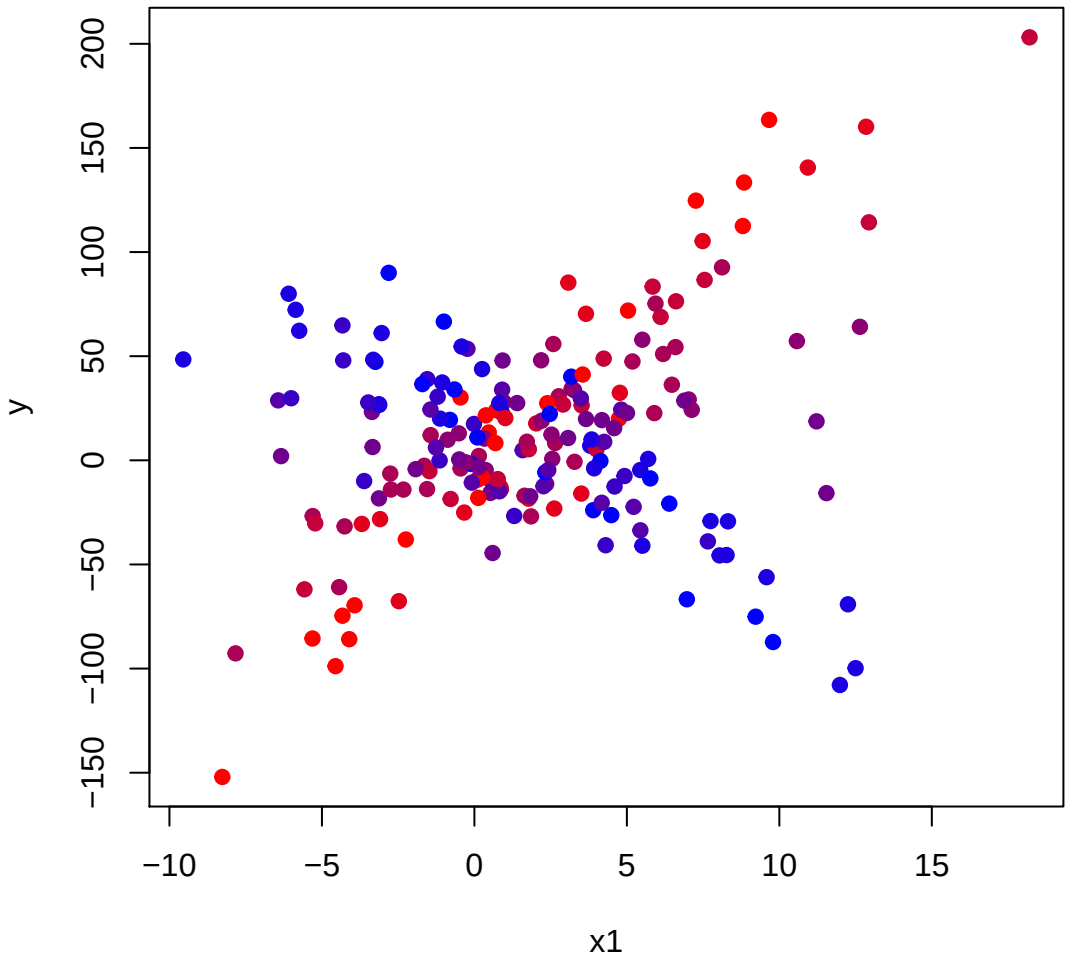
```
eff.dat <- data.frame(x2.sim, eff.x1)
eff.dat
```

```
##    x2.sim      eff.x1
## 1      -5  17.7837936
## 2      -4  14.7454971
## 3      -3  11.7072007
## 4      -2   8.6689042
## 5      -1   5.6306077
## 6       0   2.5923113
## 7       1  -0.4459852
## 8       2  -3.4842817
## 9       3  -6.5225781
## 10      4  -9.5608746
## 11      5 -12.5991711
```

The object `eff.x1` is now the coefficient of x_1 at the respective levels of x_2. x_2 in this context is also called the "moderator" or "moderating variable" because it moderates the effect of x_1. You can see that x_1 exhibits a positive effect on y when x_2 is approximately smaller than 1, at which point the effect of x_1 on y turns negative.

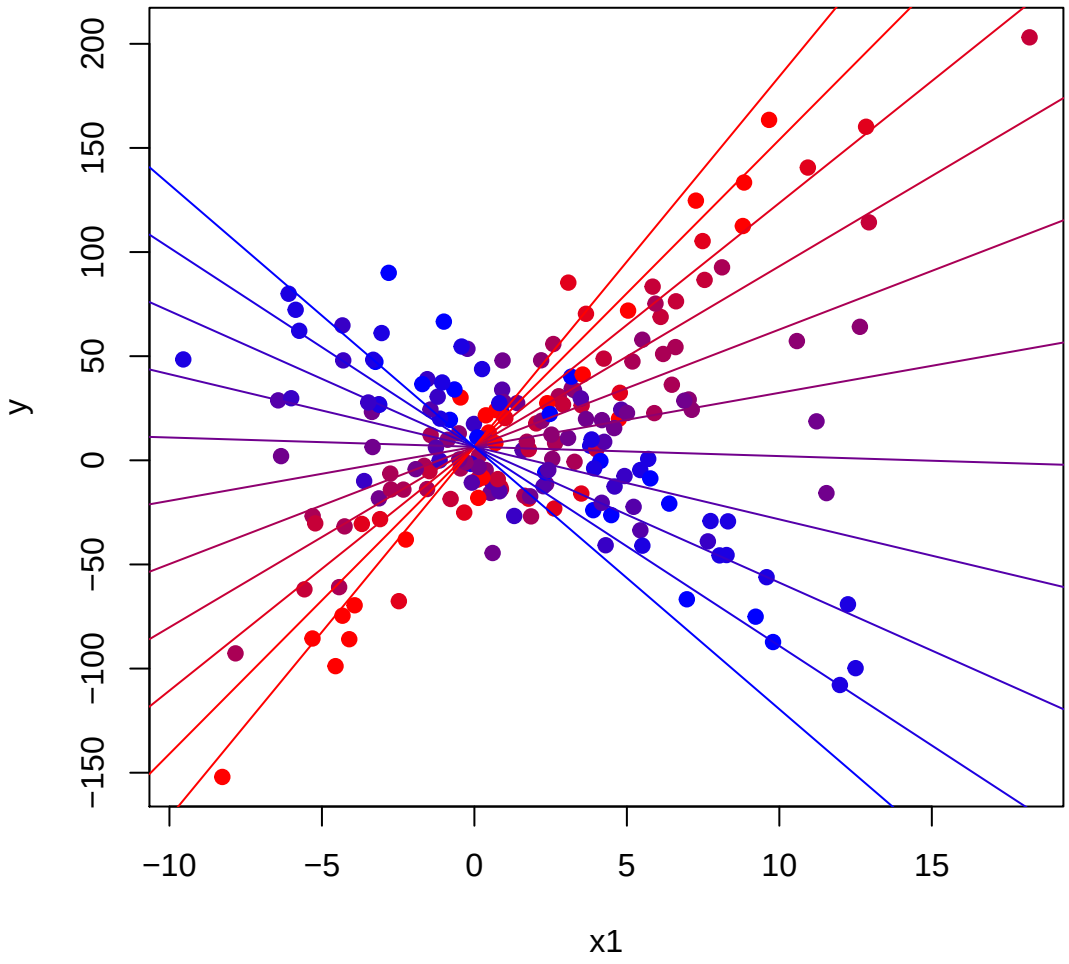
You can now visualize this, just as you did in the binary-continuous interaction above. Because x_2, the moderating variable, is now continuous, there is an (almost) infinite number of separate regression lines for x_1 and y: each individual value of x_2 creates a separate regression line (with a separate slope coefficient). The following scatterplot plots observations against their values on x_1 and 7 and colors them based on their value of x_2, as in the previous example.

```
rbPal <- colorRampPalette(c("red", "blue"))
sim.dat2$x2.col <- rbPal(10)[as.numeric(cut(sim.dat2$x2,breaks = 10))]
plot(x = sim.dat2$x1, y = sim.dat2$y, col = sim.dat2$x2.col,
     pch = 19, xlab = "x1", ylab = "y")
```



Then, I'm adding 10 regression lines for the 10 values of x_2 that I just calculated. The lines are also colored according to the values of x_2.

```
eff.dat$x2.col <- rbPal(10)[as.numeric(cut(eff.dat$x2,breaks = 10))]
plot(x = sim.dat2$x1, y = sim.dat2$y, col = sim.dat2$x2.col,
     pch = 19, xlab = "x1", ylab = "y")
apply(eff.dat, 1, function(x) abline(a = coef(mod.sim3)[1], b = x[2], col = x[3]))
```




```
## NULL
```

You can see that at low values of x_2 (red), the relationship between x_1 and y is positive (upward lines), whereas at higher values of x_2 (blue), the relationship is negative (downward lines).

Obtaining standard errors for marginal effects

As usual, you should evaluate the variance around coefficients. The online appendix to Brambor et al. (2006) gives you the formula for this variance:

For the variance of the marginal/conditional effect of x_1, use this equation:

$$\text{Var} = \text{Var}(\beta_1) + x_2^2 \times \text{Var}(\beta_3) + 2 \times x_2 \times \text{Cov}(\beta_1, \beta_3)$$

Note that this equation makes use of the *covariance* of estimates. You learned about variance and covariance on Day 10, and you can access the variance-covariance matrix with the `vcov()` function. You can then extract cells of the matrix by using the familiar square brackets, where the first number stands for rows and the second for columns.

```
vcov(mod.sim3)
```

```
##                (Intercept)            x1            x2         x1:x2
## (Intercept)  2.4277430885 -0.1831417946 -0.0009989293 -0.0018730849
## x1          -0.1831417946  0.0935660448 -0.0018849223  0.0007608924
## x2          -0.0009989293 -0.0018849223  0.2835933457 -0.0200800974
## x1:x2       -0.0018730849  0.0007608924 -0.0200800974  0.0103373423
```

```
vcov(mod.sim3)[1, 1]
```

```
## [1] 2.427743
```

Now, you can obtain the standard error of the respective coefficients by taking the square root of the variance.

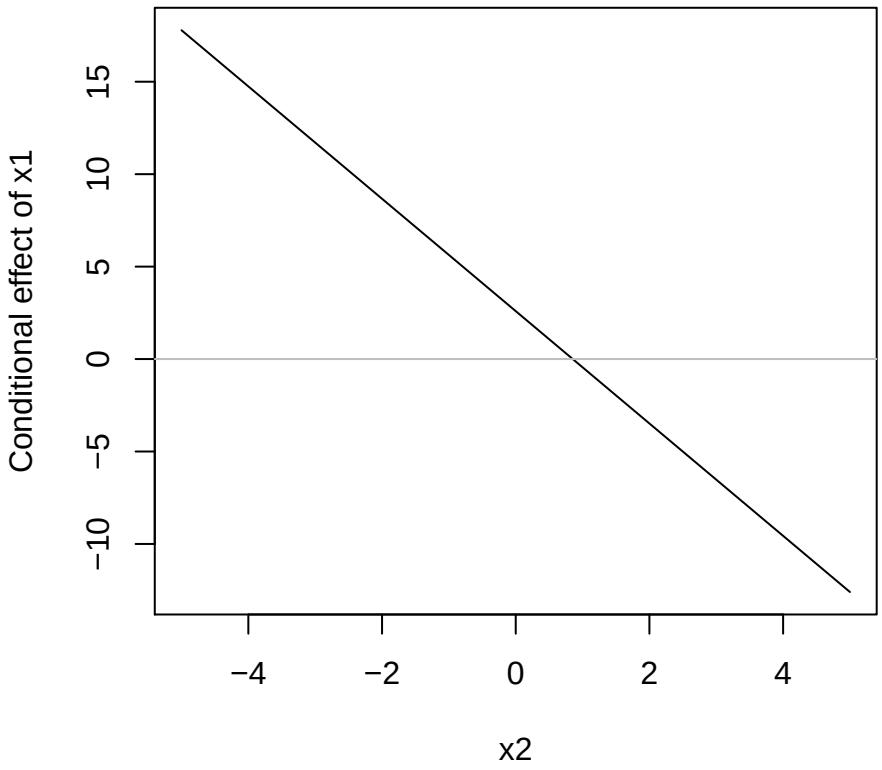
```
eff.dat$se.eff.x1 <- sqrt(vcov(mod.sim3)[2, 2] +
                            eff.dat$x2.sim^2 * vcov(mod.sim3)[4, 4] +
                            2 * eff.dat$x2.sim * vcov(mod.sim3)[2, 4])
eff.dat
```

```
##    x2.sim      eff.x1  x2.col se.eff.x1
## 1      -5  17.7837936 #FF0000 0.5868481
## 2      -4  14.7454971 #FF0000 0.5028682
## 3      -3  11.7072007 #E2001C 0.4266577
## 4      -2   8.6689042 #C60038 0.3631416
## 5      -1   5.6306077 #AA0055 0.3199713
## 6       0   2.5923113 #8D0071 0.3058857
## 7       1  -0.4459852 #71008D 0.3246924
## 8       2  -3.4842817 #5500AA 0.3714283
## 9       3  -6.5225781 #3800C6 0.4372270
## 10      4  -9.5608746 #1C00E2 0.5148307
## 11      5 -12.5991711 #0000FF 0.5996737
```

Plotting the marginal/conditional effect

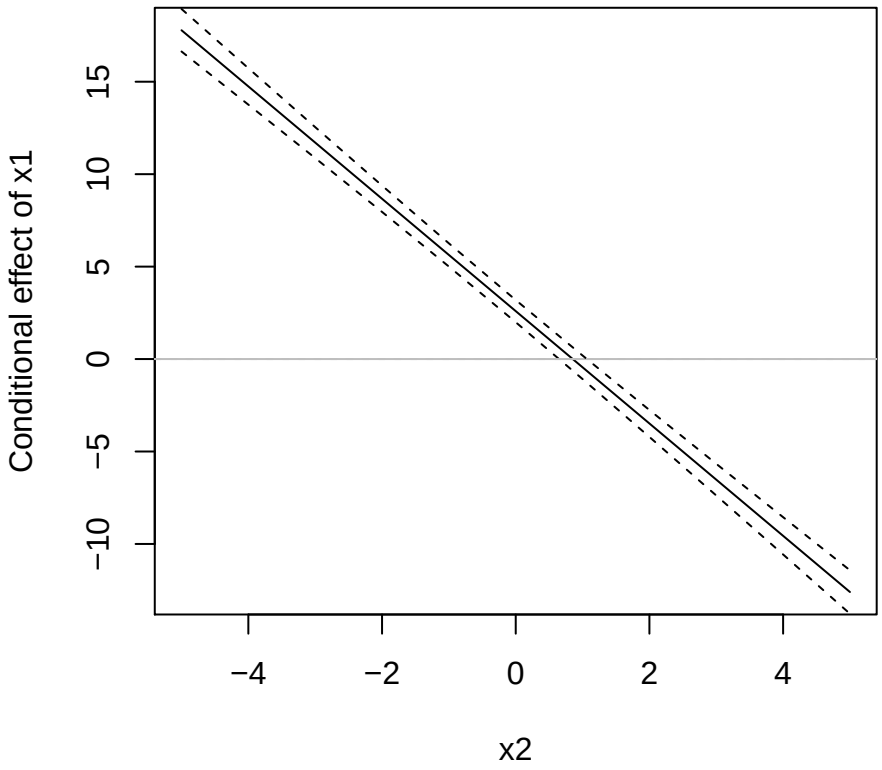
It is often more useful to plot the marginal effect of a variable across the range of the second variable in the interaction, rather than plotting separate regression lines as I did above. I strongly recommend this approach. You already have the tools to do this:

```
plot(x = eff.dat$x2.sim, y = eff.dat$eff.x1, type = "l", xlab = "x2",
     ylab = "Conditional effect of x1")
abline(h = 0, col = "grey")
```



The solid line shows the conditional effect of x_1 across the range of x_2. You can add 95% confidence intervals by plotting lines representing the conditional effect $\pm 1.96 \times$ SE.

```
plot(x = eff.dat$x2.sim, y = eff.dat$eff.x1, type = "l", xlab = "x2",
     ylab = "Conditional effect of x1")
abline(h = 0, col = "grey")
lines(x = eff.dat$x2.sim, y = eff.dat$eff.x1 + 1.96 * eff.dat$se.eff.x1, lty = "dashed")
lines(x = eff.dat$x2.sim, y = eff.dat$eff.x1 - 1.96 * eff.dat$se.eff.x1, lty = "dashed")
abline(h = 0, lty = 2, col = "grey")
```



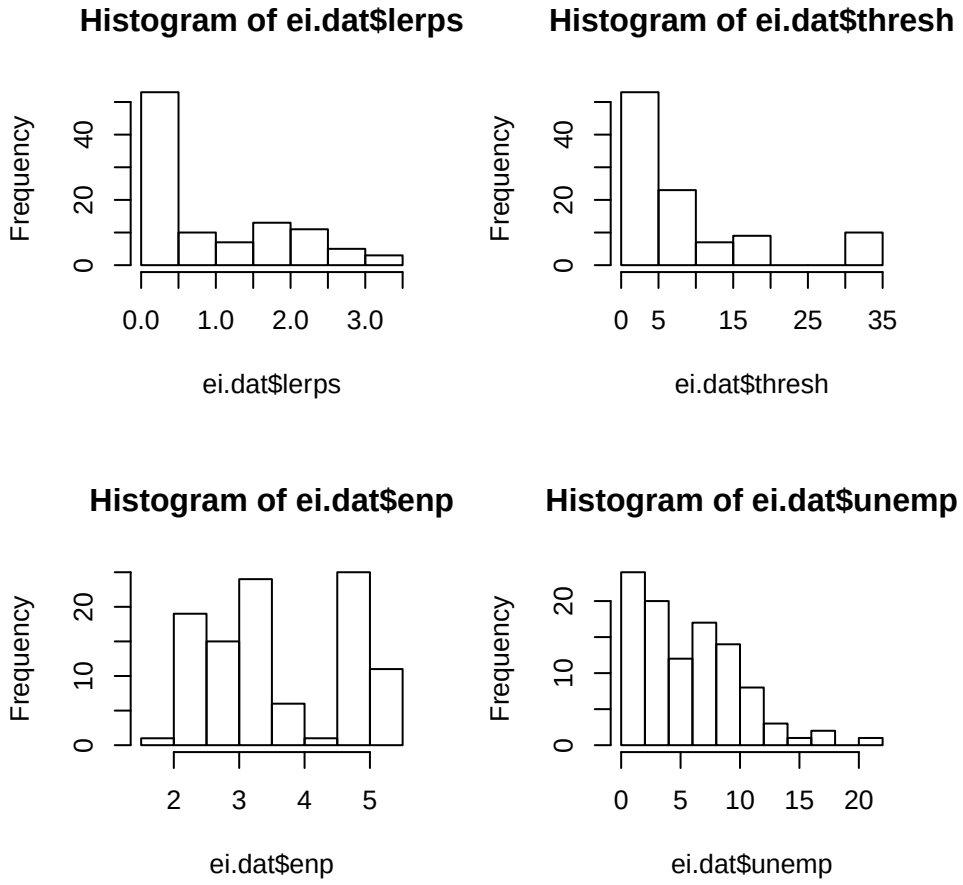
Example 3: "Electoral Institutions, Unemployment and Extreme Right Parties: A Correction."

The final example uses real-world data on the elextoral success of extreme right-wing parties in 16 countries over 102 elections. These data were used in Matt Golder's *BJPS* article "Electoral Institutions, Unemployment and Extreme Right Parties: A Correction." (see Brambor et al. 2006 for the full citation).

```
ei.dat <- read.csv("http://www.jkarreth.net/files/golder.2003.csv")
summary(ei.dat)
```

```
##     country        year          unemp            enp
##  denm   :11   Min.   :1970   Min.   : 0.300   Min.   :1.72
##  aust   : 7   1st Qu.:1975   1st Qu.: 2.125   1st Qu.:2.79
##  belg   : 7   Median :1980   Median : 5.300   Median :3.40
##  gree   : 7   Mean   :1980   Mean   : 5.811   Mean   :3.59
##  neth   : 7   3rd Qu.:1985   3rd Qu.: 8.375   3rd Qu.:4.63
##  port   : 7   Max.   :1990   Max.   :20.800   Max.   :5.10
##  (Other):56
##      lerps            thresh
##  Min.   :0.0000   Min.   : 0.670
##  1st Qu.:0.0000   1st Qu.: 2.600
##  Median :0.4700   Median : 5.000
##  Mean   :0.9208   Mean   : 8.807
##  3rd Qu.:1.8284   3rd Qu.: 9.875
##  Max.   :3.3142   Max.   :35.000
##
```

```r
par(mfrow = c(2, 2))
hist(ei.dat$lerps)
hist(ei.dat$thresh)
hist(ei.dat$enp)
hist(ei.dat$unemp)
```




```r
par(mfrow = c(1, 1))
```

Variable	Description
`lerps`	Log of extreme right percentage support + 1
`thresh`	Effective threshold of representation and exclusion in the political system
`enp`	Effective number of parties
`unemp`	Unemployment rate
`country`	Country name
`year`	Election year

This study aims to determine the relationship between electoral support for extreme right-wing parties (the outcome variable), the threshold for representation in the political system (the first predictor), and the effective number of parties (the second predictor). The author hypothesized that the effect of representation thresholds might vary depending on the effective number of parties. The unemployment rate at the time of the election is a control variable. Because elections in the same country might be subject to idiosyncratic dynamics, the author includes a dummy variable for each country (cf. our discussion on Days 9 and 10).

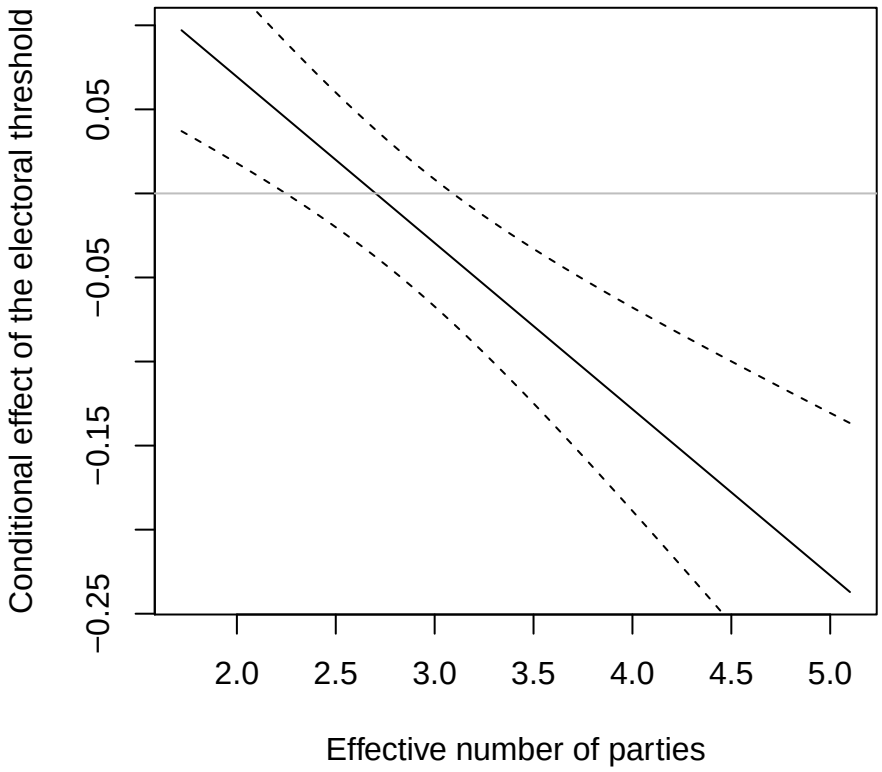
After fitting the model, I display the results using the familiar `screenreg()` function from the "texreg" package. I make use of the `omit.coef` option to avoid printing of the coefficients for each country dummy variable.

```
# Model 3
m3 <- lm(lerps ~ thresh + enp + thresh * enp + unemp + factor(country), data = ei.dat)
library(texreg)
screenreg(list(m3), omit.coef = "country")
```

```
##
## ========================
##               Model 1
## ------------------------
## (Intercept)    -0.97
##                (1.10)
## thresh          0.27 ***
##                (0.06)
## enp             1.16 **
##                (0.43)
## unemp           0.07 ***
##                (0.02)
## thresh:enp     -0.10 ***
##                (0.02)
## ------------------------
## R^2             0.85
## Adj. R^2        0.82
## Num. obs.     102
## ========================
## *** p < 0.001, ** p < 0.01, * p < 0.05
```

Rather than trying to interpret these results from the regression table, I calculate the marginal effect of electoral thresholds across the range of the effective number of parties. This step is exactly analogous to what I did with simulated data above.

```
thresh_sim <- seq(from = min(ei.dat$thresh), to = max(ei.dat$thresh), length.out = 50)
enp_sim <- seq(from = min(ei.dat$enp), to = max(ei.dat$enp), length.out = 50)
eff_thresh <- coef(m3)[2] + coef(m3)[20] * enp_sim
eff_thresh_se <- sqrt(vcov(m3)[2, 2] + enp_sim^2 * vcov(m3)[20, 20] + 2 * enp_sim * vcov(m3)[2, 20])
plot(x = enp_sim, y = eff_thresh, type = "l",
     xlab = "Effective number of parties",
     ylab = "Conditional effect of the electoral threshold")
lines(x = enp_sim, y = eff_thresh + 1.96 * eff_thresh_se, lty = "dashed")
lines(x = enp_sim, y = eff_thresh - 1.96 * eff_thresh_se, lty = "dashed")
abline(h = 0, col = "grey")
```



3. What do you conclude from this figure? What do these results say about the author's hypothesis?

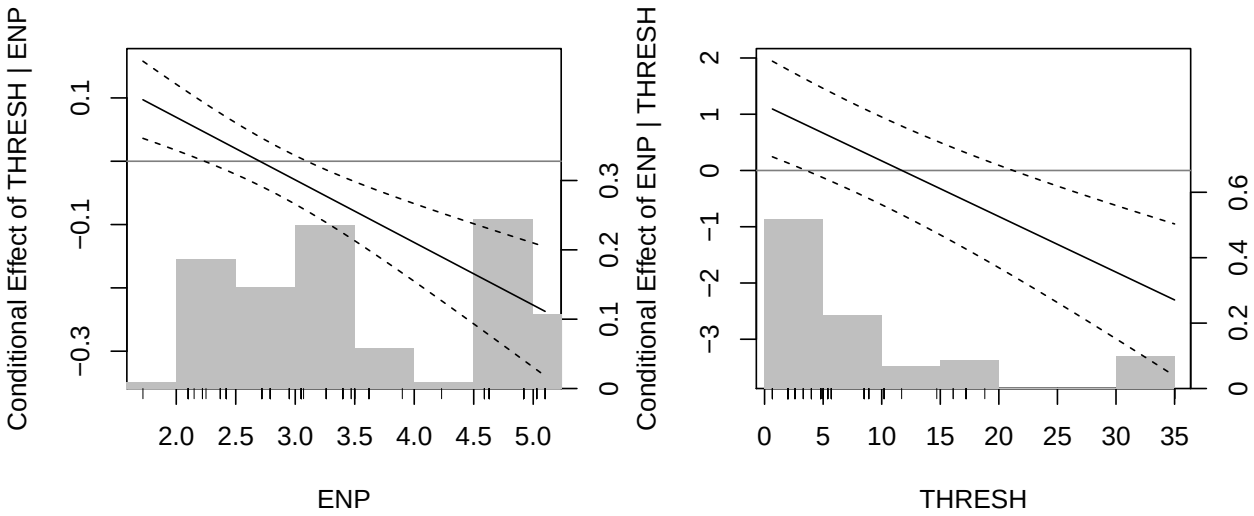
Canned R functions to plot marginal effects in OLS

Rather than calculating marginal effects by hand, you might want to use R functions to plot marginal effects with less effort.

`DAintfun2()`

One good option is Dave Armstrong's `DAintfun2()` function, which is part of his "DAMisc" package. You need to first install the package, then load it, and then specify the function with at least the following arguments:

```
# install.packages("DAMisc")
library(DAMisc)
DAintfun2(m3, varnames = c("thresh", "enp"),
                  rug = TRUE, hist = TRUE)
```



`DAintfun2()` by default plots both the marginal effect of the first variable as well as that of the second variable. It also allows you to include an indicator of the actual distribution of the variables, via a histogram or a rug plot. With these supplementary plots, you can immediately see whether the estimated effect of a variable corresponds to real observations at that value of the moderating variable.

`ggintfun()`

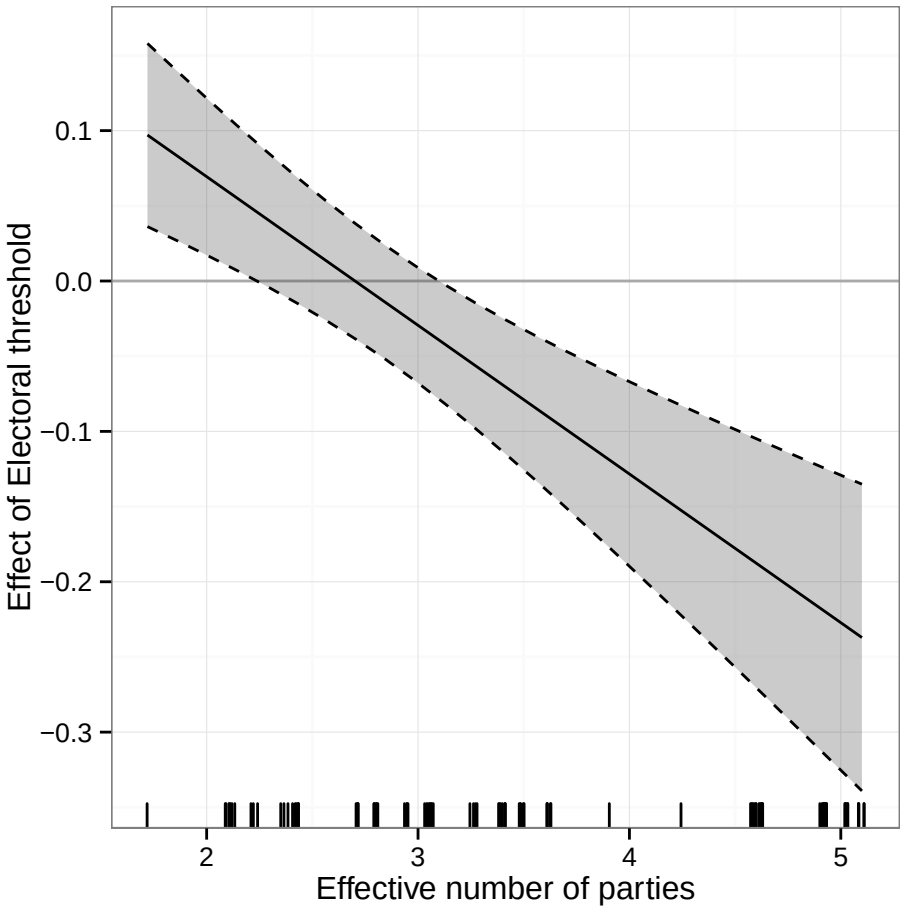
A second function is `ggintfun()`, which I've written. I prefer working with ggplot and wanted to have a customizeable version of a marginal effects plot; otherwise, `DAintfun2()`does everything I need. `ggintfun()` uses the "ggplot2" package and you need to download it from my GitHub repository. This requires three extra steps:

- Install the "devtools" package (only once).
- Load the "devtools" package (every time you want to use `ggintfun()`).
- Read the `ggintfun()` function into R using the `source_url()` function and the URL of the raw R code for `ggintfun()` on my GitHub site (every time you want to use `ggintfun()`).

```
# install.packages("devtools")
library(devtools)
source_url("https://raw.githubusercontent.com/jkarreth/JKmisc/master/ggintfun.R")
```

The function then produces this plot, which shows the exact same information as your hand-crafted marginal effects plot and the left panel of the `DAintfun2())` plot from above.

```
ggintfun(m3, varnames = c("thresh", "enp"),
         varlabs = c("Electoral threshold", "Effective number of parties"),
         title = FALSE, rug = TRUE,
         twoways = FALSE)
```


Effect of Electoral threshold
0.1
0.0
−0.1
−0.2
−0.3
2
3
4
5
Effective number of parties