Lecture 8.4: Multiple regression part IV: finishing up interactions

* *This lecture is based on chapter 7 of Statistical Rethinking by Richard McElreath.*

As always, we need to load some packages and set some options prior to running any models:

```r
library(rstan)
library(shinystan)
library(car)
library(xtable)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
source("../utilityFunctions.R")
```

Last lecture, I introduced interaction terms using the GDP dataset. * Recall we were interested in predicting `log(GDP)` in African/non-African countries as a function of terrain ruggedness.

```r
rugged <- read.csv("RuggedGDP.csv")
rugged <- rugged[order(rugged$rugged),]
afr <- rugged[rugged$africa == 1, ] # African dataset
nafr <- rugged[rugged$africa == 0, ] # non-African dataset
afr <- afr[order(afr$rugged),]
nafr <- nafr[order(nafr$rugged),]
```

We wanted the relationship between *obs* and R to vary as a function of A.

$$\begin{aligned} obs_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \gamma_i R_i + \beta_A A_i \\ \gamma_i &= \beta_R + \beta_{AR} A_i \end{aligned}$$

Our Bayesian model had two linear models in it:

1. The first line is the same Gaussian likelihood we all know and love.
2. The second line is the same additive definition of μ_i.
3. The third line uses γ as a placeholder for our new linear function that defines the slope between `log(GDP)` and `rugged`.
    - γ_i is the linear interaction effect of ruggedness and African nations

γ_i explicitly models the hypothesis that the slope between GDP and ruggedness is *conditional* on whether a nation is on the African continent with β_{AR} describing the strength of that dependence.

- If $\beta_{AR} = 0$, then we get our original likelihood function back.
  - For any nation not in Africa, $A_i = 0$ and so β_{AR} has no effect.
- If $\beta_{AR} > 1$, African nations have a more positive slope between GDP and ruggedness.
- if $\beta_{AR} < 1$, African nations have a more negative slope

When we created our `Stan` model last lecture, we used the conventional notation by substituting in γ_i:

$$\begin{aligned}\gamma_i &= \beta_R + \beta_{AR} A_i \\ \mu_i &= \alpha + \gamma_i R_i + \beta_A A_i \\ &= \alpha + (\beta_R + \beta_{AR} A_i) R_i + \beta_A A_i \\ &= \alpha + \beta_R R_i + \beta_{AR} A_i R_i + \beta_A A_i.\end{aligned}$$

This form is much easier to code in `Stan`.

I left the first model for you to try as a homework because 1) it is more explicit; 2) it will be easier to interpret the results; and 3) understanding this form will help in understanding (and building) more complex hierarchical models later.

- There are about 5 different ways to code the first model. The easiest is to do the following:

```
data {
  int<lower=0> nObs;
  vector[nObs] obs;
  vector[nObs] R;
  vector[nObs] A;
  real<lower=0> aMu;      // mean of prior alpha
  real<lower=0> aSD;      // SD of prior alpha
  real<lower=0> bMu;      // mean of prior betas
  real<lower=0> bSD;      // SD of prior beta
  real<lower=0> sigmaSD;  // scale for sigma
}

parameters {
  real alpha;
  real betaR;
  real betaA;
  real betaAR;
  real<lower=0> sigma;
}

transformed parameters {
  vector[nObs] mu;
  vector[nObs] gamma;

  gamma = betaR + betaAR*A;
  // elementwise multiplication (.*)!
  mu = alpha + gamma .* R + betaA*A;
}

model {
  alpha ~ normal(aMu, aSD);
  betaR ~  normal(bMu, bSD);
  betaA ~  normal(bMu, bSD);
  betaAR ~  normal(bMu, bSD);
  sigma ~ cauchy(0, sigmaSD);

  obs ~ normal(mu, sigma);
}
```

Could also do the following:

```
transformed parameters {
  vector[nObs] mu;
  vector[nObs] gamma;

  gamma = betaR + betaAR*A;
  mu = alpha + to_vector(gamma * R') + betaA*A;
}
```

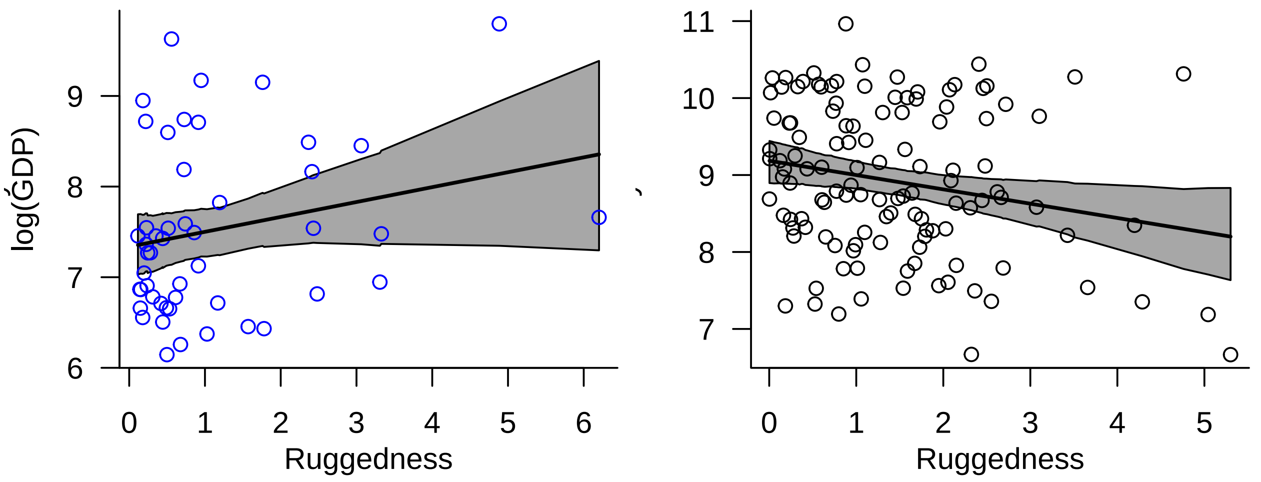
The ' symbol means transpose and turns a `nObs` × 1 `vector` into a 1 × `nObs` vector for proper matrix multiplication (and results in a `nObs` × 1) output matrix).

- Through the vagaries of Stan, you can't add together matrices and vectors. Therefore we coerce `gamma` × `R` into a vector using `to_vector()`.
- Alternatively we could just define `R` as a `row_vector` in the `data` block and skip the transpose.

Regardless of which model we use, we get the same results. If we run either model

```
# African linear regression
X <- model.matrix(~rugged*africa, data=rugged)[,2:4]
intDat <- list(nObs=nrow(rugged), obs=log(rugged$GDP),
  R=X[,"rugged"], A=X[,"africa"], aMu=0, aSD=20, bMu=0, bSD=1, sigmaSD=10)

intxnMod <- stan(file="intrxnMod8.4.stan", data=intDat, iter=2000,
 chains=4, seed=867.5309)
mu <- as.matrix(intxnMod, "mu")
muHDI <- apply(mu, 2, HDI, credMass=0.95)
muMn <- colMeans(mu)
```



we find a positive relationship between `log(GDP)` and `ruggedness` for African nations and a negative relationship for non-African nations.

Interpreting interaction terms

Interpreting interaction terms is not easy

- Plotting out the results is usually best for model inference.
- However, there are times when interpretation of the parameter estimates themselves is of value.

There are two reasons why interpreting tables of parameter estimates from models with interaction terms is challenging:

1. *Adding an interaction to the model changes parameter meanings.* Usually, the distribution of a “main effect” coefficent in an interaction model can not be directly compared to a term of the same name in a non-interaction model.
2. *Interpreting tables of interaction effects require thinking about parameter covariance.* This is a lot harder when the influence of a predictor depends on multiple parameters.

Adding an interaction to the model changes parameter meanings:

In a simple linear regression without interaction terms, each predictor variable is independent and directly measures that variable’s influence.

- Not so for models with interaction terms

If we look at our likelihood function again,

$$\begin{aligned} obs_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \gamma_i R_i + \beta_A A_i \\ \gamma_i &= \beta_R + \beta_{AR} A_i \end{aligned}$$

a change in μ_i is dependent on a unit change in R_i is governed by γ_i.

- γ_i is a function of β_R, β_{AR}, and A_i; we need to know all three to interpret the effect of R_i on the outcome.
  - Only when $A_i = 0$ can we interpret the slope β_R because then $\gamma_i = \beta_R$.

Practically, this means interpreting tables of the estimates requires some math. If we want to estimate the effect of ruggedness on `log(GDP)` within Africa

```
round(summary(intxnMod, pars=c("alpha", "betaR", "betaA",
  "betaAR"), probs = c(0.025, 0.5, 0.975))$summary,2)
```

```
        mean se_mean   sd  2.5%   50% 97.5%   n_eff Rhat
alpha   9.19    0.00 0.14  8.90  9.19  9.46 1397.29    1
betaR  -0.19    0.00 0.08 -0.34 -0.19 -0.03 1359.30    1
betaA  -1.85    0.01 0.22 -2.30 -1.85 -1.41 1558.10    1
betaAR  0.35    0.00 0.13  0.10  0.35  0.61 1704.45    1
```

If we want to estimate the effect of ruggedness on `log(GDP)` within Africa,

$$\gamma = \beta_R + \beta_{AR}(1) = -0.19 + 0.35 = 0.16.$$

Outside of Africa,

$$\gamma = \beta_R + \beta_{AR}(0) = -0.19 = -0.19,$$

so the relationship is essentially reversed.

Interpreting tables of interaction effects require thinking about parameter covariance:

But those are only point estimates of the marginal values. If we really want to compare the effects between African and non-African countries, we have to consider the whole posterior distribution:

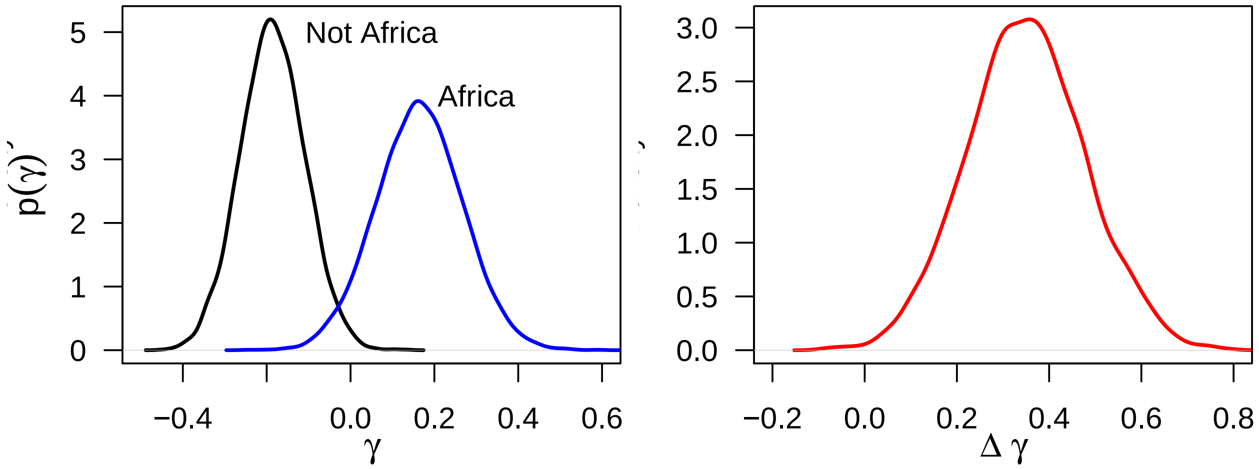
```
slopes <- as.matrix(intxnMod, pars=c("betaR", "betaAR"))
gammaA <- slopes[,"betaR"] + slopes[,"betaAR"]*1
gammaNA <- slopes[,"betaR"] + slopes[,"betaAR"]*0
mean(gammaA)
```

```
[1] 0.1643487
```

```
mean(gammaNA)
```

```
[1] -0.1858786
```

The means are almost identical to those above.



But we can also plot the full distributions together, or the diference in the distributions. Note that the proportion of the differences less than zero is only 0.003

- much less than the overlap of the marginal distributions suggests.