# Lecture 8.2: Multiple regression part II: multicollinearity

** This lecture is based on chapter 5 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)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
source("../utilityFunctions.R")
```

In lecture 8.1, we discussed why we might want multiple predictors in the same regression model and learned how to set up these models in `Stan`.

- We also learned about how to visualize multiple regressions using things like counterfactual plots.

As a recap, the model was specified as:

$$
\begin{aligned}
obs_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \sum_{j=1}^{n} \beta_j x_{ji} \\
&= \alpha + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_n x_{ni} \\
\alpha &\sim \text{Normal}(a_\mu, a_{SD}) \\
\beta_j &\sim \text{Normal}(b_\mu, b_{SD}) \\
\sigma &\sim \text{Cauchy}^+(0, \sigma_{SD})
\end{aligned}
\tag{1}
$$

If multiple predictors are the bee's knees and improve model inference, why not add all possible predictors? There are at least three good reasons:

1. *Multicollinearity:* a strong correlation between two or more predictor variables.
2. *Post-treatment bias:* mistaken inferences arising from variables that are consequences of other variables
   - opposite of *omitted variable bias* which we discussed last time.
3. *Overfitting:* We will talk about this in a few weeks.

# Multicollinearity

The consequence of multicollinearity (as we will see) is that the posterior distribution will suggest a huge range of plausable parameter values , from small associations to enormous ones.

- occurs even if all variables truely have strong associations with the response variable.

## Simulation of multicollinearity

This example is rather artificial, but may make sense to the anthropologists in the class.

*Example:* Prediction of an individual's height using leg length as predictors. We would expect that height is postively associated with leg length, but strange(r) things happen if we put both leg lengths into the model!

We will simulate heights and leg lengths for $N = 100$ individuals in 3 steps:

1. First we simulate heights from a Gaussian distribution

```
N <- 100                  # number of individuals
height <- rnorm(N, 10, 2) # simulated heights
```

2. Then we simulate leg lengths as proportions (0.4–0.5) of height

```
legProp <- runif(N, 0.4, 0.5) # leg as proportion of height
```

3. Each leg gets measurement/developmental error tacked on so the right and left legs are not exactly equal in length, as is typical

```
leftLeg <- height*legProp + rnorm(N, 0, 0.02) # sim left leg
rightLeg <- height*legProp + rnorm(N, 0, 0.02)# sim right leg
```

Before fitting the model, what should we expect?

- On average, leg length should be 45% of an individual's height for these data.
- $\beta_{leg}$ measures the relationship with leg length and height.
- Therefore, $\beta_{leg}$ should equal the average height (10) divided by 45% of the average height (4.5) $\Rightarrow 10/4.5 \approx 2.2$.

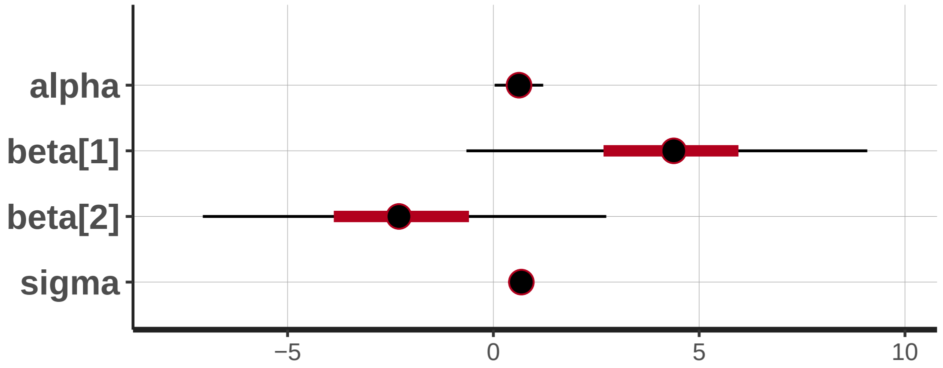
Using a multiple regression model similar to that from last time (`multMod8.2.stan`):

```
data {
  int<lower=0> nObs;
  int<lower=0> nVar;      // no. vars
  vector[nObs] obs;
  matrix[nObs, nVar] X;
  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;
  vector[nVar] beta;
  real<lower=0> sigma;
}

transformed parameters {
  // can be useful for plotting purposes
  vector[nObs] mu;
  mu = alpha + X*beta;
}

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

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

let's see what happens:

```
dat <- list(nObs=N, nVar=2, obs=height, X=cbind(leftLeg, rightLeg),
  aMu=10, aSD=10, bMu=2, bSD=10, sigmaSD=10)

legsMod <- stan(file="multMod8.2.stan", data=dat, iter=2000,
 chains=4, seed=867.5309, pars="mu", include=FALSE)
```

```
plotLegs <- plot(legsMod, pars=c("alpha", "beta", "sigma"), ci_level=0.5)
plotLegs + theme(text=element_text(family="ArialMT"))
```



```
round(summary(legsMod, pars=c("alpha", "beta", "sigma"),
  probs = c(0.025, 0.5, 0.975))$summary,2)
```

```
          mean se_mean   sd  2.5%   50% 97.5%   n_eff Rhat
alpha     0.63    0.01 0.31  0.03  0.62  1.21 2175.94    1
beta[1]   4.33    0.07 2.46 -0.65  4.39  9.08 1377.51    1
beta[2] -2.25    0.07 2.47 -7.06 -2.30  2.74 1375.39    1
sigma     0.68    0.00 0.05  0.59  0.68  0.79 2107.93    1
```

Both the plot and printed output show that something has gone wrong. The means are weird and the uncertainties are huge!

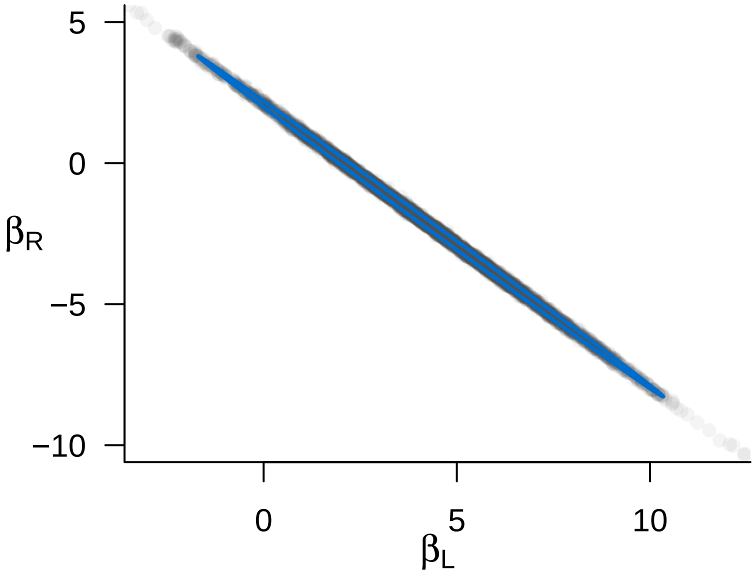
- If both legs have almost identical lengths and height is so strongly correlated with leg length, why is this posterior distribution so odd?

The model did exactly what it should have, and the posterior distribution is exactly what we asked for.

- A multiple regression answers: *What is the value of knowing each predictor after knowing all of the other predictors?*
  - i.e., *What can we say about height knowing the right (left) leg length after already knowing the left (right) leg length?*

It will help to look at the bivariate posterior distribution of $\beta_{Left}$ and $\beta_{Right}$:

```
col <- "#50505010"

legs <- as.matrix(legsMod, pars="beta")
# Plot alpha and beta
plot(legs, pch=16, col=col, las=1, xlim=c(-3,12),
  ylim=c(-10,5), bty="l", ann=FALSE)
dataEllipse(legs,level=c(0.95), add=TRUE, labels=FALSE,
  plot.points=FALSE, center.pch=FALSE, col=c(col,"#006DCC"))

mtext(text=expression(beta[R]), side=2, line=2.1, cex=1.2, las=1)
mtext(text=expression(beta[L]), side=1, line=2, cex=1.2)
```



The posterior distribution of $\beta_L$ and $beta_R$ is highly correlated ($\rho =$ -0.9996).

- when $\beta_L$ is large, $\beta_R$ must be small.
- Both leg variables contain almost the same information.
  - including both of them in the model results in an almost infinite number of combinations of $\beta_L$ & $\beta_R$ that produce the same predictions.

We can think of this as approximating the following likelihood:

$$\begin{aligned} y_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i \end{aligned}$$

where $y$ is the outcome (e.g., `height`) and x is a single predictor (e.g., `leg lengths`) used twice.

From the computer's point of view, this is:

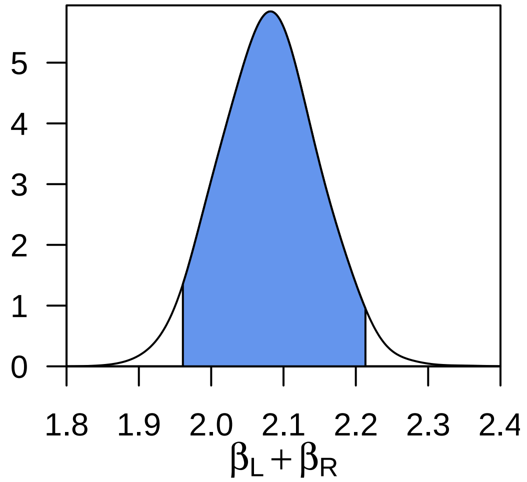
$$\mu_i = \alpha + (\beta_1 + \beta_2)x_i.$$

We cannot separate out $\beta_L$ and $\beta_R$ because they do not separately influence the mean $\mu$.

- Only their sum $\beta_L + \beta_R$ does.
- This means there are infinite combinations of $\beta_L$ & $\beta_R$ that make the sum close to the actual association of `leg length` with `height`.

And our model has done exactly that. If we compute the posterior distribution of the $\beta's$ sum and plot it:

```
par(mar=c(3,3.2,0.1,0.5))
plotInterval(rowSums(legs), HDI=TRUE, interval=0.95, xlims=c(1.8, 2.4),
  col="cornflowerblue", yOffset=0.1)
mtext(expression(paste(beta[L] + beta[R])), side=1, line=2.1, cex=1.2)
```



The posterior mean is a bit over 2 and the SD (0.07) is much less than the SD for either $\beta$ ($\approx 2.4$).

If we fit a regression with just one of the variables (`uniMod.stan`), we get about the same posterior mean:

```
dat1 <- list(nObs=N, nVar=2, obs=height, xvar=leftLeg,
  aMu=10, aSD=10, bMu=2, bSD=10, sigmaSD=10)

legMod <- stan(file="uniMod.stan", data=dat1, iter=2000, chains=4,
  seed=2, pars=c("alpha","beta", "sigma"))

round(summary(legMod, probs = c(0.025, 0.5, 0.975))$summary,2)
```

```
        mean se_mean   sd   2.5%    50%  97.5%   n_eff Rhat
alpha   0.58    0.01 0.30   0.00   0.57   1.15 1419.31 1.01
beta    2.09    0.00 0.07   1.96   2.09   2.21 1391.97 1.01
sigma   0.68    0.00 0.05   0.59   0.68   0.79 1829.46 1.00
lp__  -11.57    0.03 1.23 -14.80 -11.25 -10.16 1239.30 1.00
```

**So how large does a correlation have to become before we should start worrying?**

This question will depend on the situation and priors, but what matters is not just the correlation between pairs of variables.

- It is the correlation that remains after accounting for other predictors.

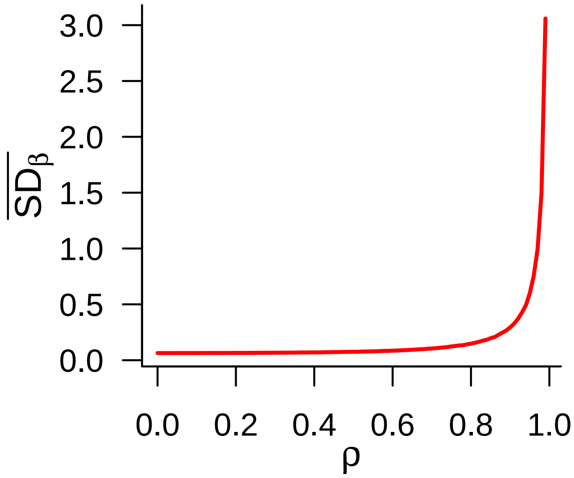
However, with only two predictor variables, we can do a little simulation experiment to address the correlation question directly.

1. Suppose we only have height and the left leg length.
2. We can make a random predictor $(x)$ that is correlated with $leg_L$ at a predetermined $\rho$.
3. We will fit a regression model that predicts height using $leg_L$ and $x$.
4. Record the SE of the estimated effect of $leg_L$
5. Repeat multiple iterations for many values of $\rho$.

The following code will do exactly that:

```r
# function to simulate collinearity for 1 iter
collSim <- function (rho=0.9) {
  x <- rnorm(N, mean=rho*leftLeg, sd=(1-rho^2)*var(leftLeg))
  m <- lm(height ~ leftLeg + x)
  return(sqrt(diag(vcov(m))[2]))
}

# function to replicate collSim iter times for a given rho
repColSim <- function (rho=0.9, iter=100) {
  stdev <- replicate(iter, collSim(rho))
  return(mean(stdev))
}

rhoSeq <- seq(0, 0.99, by=0.01)  # sequence of correlations
stddev <- sapply(rhoSeq, function(z) {repColSim(rho=z, iter=100)})
```

```r
plot(rhoSeq, stddev, type="l", bty="l", ann=FALSE, las=1, col="red", lwd=2)
mtext(text=expression(bar(SD[beta])), side=2, line=2.1, cex=1.2)
mtext(text = expression(rho), side=1, line=2, cex=1.2)
```



Here the $y$ axis is the average SD across 100 replications per $\rho$ using a simulated correlated predictor variable $x$.

- When the two variables are uncorrelated, the SD of the posterior is small. As the correlation gets larger, the SD inflates.
- The inflation is also non-linear. When $\rho > 0.9$, it accelerates rapidly towards $\infty$.
    - Note that this model is essentially Bayesian and assuming flat uninformative priors. More informative priors can help mitigate some of the effects of multicollinearity.

Multicollinearity is a type of *non-identifiability*. This means the structure of the data and/or model make it impossible to estimate the values of parameters.

In our example, this is because there are an infinite number of combinations of parameters that are equally plausible.

## Post-treatment bias

Another pitfall of multiple regression arises from mistaken inferences due to including variables that are consequences of other variables.

As a motivating ecological example, suppose we have a greenhouse experiment with plants (maybe *Eucalyptus*?) and want to understand the impact of fungal treatments on growth.

- Plants are germinated and let grow for a set amount of time, and then their heights $H_0$ are measured.
- We then apply a fungal treatment to half of the replicates' soils.
- Final heights $H_1$ and the presence of soil fungus are then measured.

We have four variables of interest: initial and final heights, treatment, and the presence of fungus.

If causal inference about the effect of treatment is of interest however, we shouldn't include fungal presence because it is a post-treatment effect.

I have simulated such an example and the results are saved as `plants.csv`. The simulation code is below:

```r
# Number of plants
N <- 100
set.seed(3)
# initial heights
h0 <- rnorm(N, 10, 2)

# assign treatments and simulate growth and fungal persistence
treatment <- rep(0:1, each=N/2)
fungus <- rbinom(N, size=1, prob=0.5-treatment*0.4)
```

```
h1 <- h0 + rnorm(N, 5-3*fungus)

plants <- data.frame(h0=h0, h1=h1, treats=treatment, fungus=fungus)
write.csv(plants, "plants.csv", row.names=FALSE)
```

```
plants <- read.csv("plants.csv")
head(plants)
```

```
         h0        h1 treats fungus
1  8.076133  8.784784      0      1
2  9.414949 14.049994      0      1
3 10.517576 16.004649      0      0
4  7.695736 13.549629      0      0
5 10.391566 13.480008      0      1
6 10.060248 15.286262      0      0
```

```
obs <- plants$h1
# create a design matrix with initial height centered but not scaled
X <- cbind(as.vector(scale(plants$h0, scale=FALSE)),
  plants$treats, plants$fungus)
```

First off, this is our first example with *dummy coding*. Dummy variables encode qualitative catagories (e.g., treatment) into quantitative models.

- The effect of dummy coding is to turn variables on (1) or off (0) in each catagory.

For example, if we just created a model of the effect of treatment on final height:

$$
\begin{aligned}
h_{1i} &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_{treat} x_i
\end{aligned}
$$

the parameter $\beta_{treat}$ only influences prediction for cases where `treats` $= 1$. When `treats` $= 0$, it has no effect because the model is $\alpha + \beta_{treats} \times 0$.

- The intercept is the mean value when `treats` $= 0$.

In the full model, $\alpha$ is the average value when `treats` $= 0$, `fungus` $= 0$, and the initial height (`h0`) is at it's average (because we centered for interpretability).

```
dat <- list(nObs=nrow(plants), nVar=dim(X)[2], obs=obs, X=X,
  aMu=10, aSD=10, bMu=0, bSD=10, sigmaSD=10)

plantMod <- stan(file="multMod8.2.stan", data=dat, iter=2000,
  chains=4, seed=2, pars=c("alpha","beta", "sigma"))

round(summary(plantMod, pars=c("alpha", "beta", "sigma"),
  probs = c(0.025, 0.5, 0.975))$summary,2)
```

```
         mean se_mean   sd  2.5%   50% 97.5%   n_eff Rhat
alpha   15.34    0.01 0.22 14.89 15.35 15.77 1743.67    1
beta[1]  1.07    0.00 0.07  0.93  1.06  1.20 3661.86    1
beta[2] -0.17    0.01 0.25 -0.69 -0.16  0.33 2137.94    1
beta[3] -3.27    0.01 0.28 -3.81 -3.26 -2.71 1975.26    1
sigma    1.14    0.00 0.09  0.99  1.14  1.33 3404.67    1
```

If we look at the marginal posterior for $\beta_{treats}$ (`beta[2]`), the effect of treatment is actually negative but for all intents and purposes has a negligable effect.

The initial height (`beta[1]`) and fungal presence (`beta[3]`) have important effects. Does treatment not matter?

The problem is that fungal presence is predominantly a consequence of treatment. When we include `fungus` in the model, we are asking *Once we already know whether a plant has developed a fungal infection, does soil treatment matter?*

- The answer is no because the soil treatment affects plant growth by preventing fungal growth. If the fungus grew, qualitatively it doesn't matter whether the plants were treated or not.
- Given the experimental design, we want to know the impact of treatment on growth.
- If we exclude fungus:

```
X1 <- cbind(as.vector(scale(plants$h0, scale=FALSE)),
  plants$treats)

datNF <- list(nObs=nrow(plants), nVar=dim(X1)[2], obs=obs, X=X1,
  aMu=10, aSD=10, bMu=0, bSD=10, sigmaSD=10)

modNF <- stan(file="multMod8.2.stan", data=datNF, iter=2000,
  chains=4, seed=2, pars=c("alpha","beta", "sigma"))

round(summary(modNF, pars=c("alpha", "beta", "sigma"),
  probs = c(0.025, 0.5, 0.975))$summary,2)
```

```
         mean se_mean   sd  2.5%   50% 97.5%   n_eff Rhat
alpha   13.56    0.00 0.26 13.05 13.56 14.09 2868.38    1
beta[1]  0.97    0.00 0.11  0.76  0.97  1.18 3593.56    1
beta[2]  1.17    0.01 0.37  0.46  1.18  1.89 2955.86    1
sigma    1.83    0.00 0.13  1.59  1.82  2.11 3300.16    1
```

Now the impact of fungal treatment `beta[2]` $= 1.17$ is substantial and positive.

Including pre-treatment variables as controls (e.g., initial plant height) makes sense because otherwise they might mask causal effects of treatment.

In rigorously controlled experimental studies it is (hopefully) easy to identify pre- vs. post-treatment variables.

- But post-treatment variables can mask the effects of the treatment itself.

The problem with post-treatment variables is that sometimes (e.g., observational studies), it is hard to identify them.

- Moreover, model selection (which we will cover soon) may not help, because a model that includes both `fungus` and `treatment` fits the data better and makes better predictive inferences.
  - The full model would mislead because it is asking the wrong question, not because it is making poor predictions.