HW 3 Math 463 Spring 2022

Ian McConachie

4/20/2022

10.1: Regression with Interactions

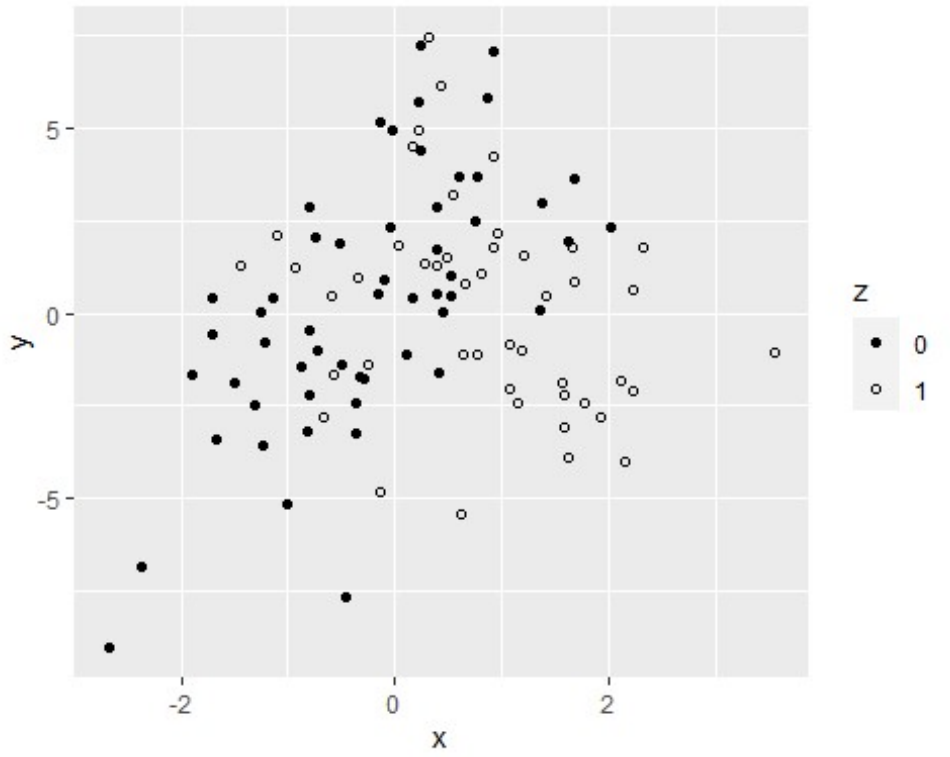
This question asks to us to simulate points from the model y = b_0 + b_1x + b_2z + b_3xz + error with a continuous predictor x and a binary predictor z, coefficients b = c(1, 2, −1, −2), and errors drawn independently from a normal distribution with mean 0 and standard deviation 3, as follows. For each data point i, first draw z_i, equally likely to take on the values 0 and 1. Then draw xi from a normal distribution with mean z_i and standard deviation 1. Then draw the error from its normal distribution and compute y_i.

The R code below simulates the data we will use for this problem.

```
size = 100
# First simulate coefficients
z = as.matrix(rbinom(n=size, size=1, prob=0.5))
x = as.matrix(rnorm(n=size, mean=z, sd=1))
error = as.matrix(rnorm(n=size, mean=0, sd=3))
# Then do linear comination to get y values
y = 1 + (2*x) + (-1*z) + (-2*(x*z)) + error
simdata = data.frame(z,x,error,y)
```

Now we can plot the data we generated above with x against y and z designated as a categorical variable that can be either 0 or 1.

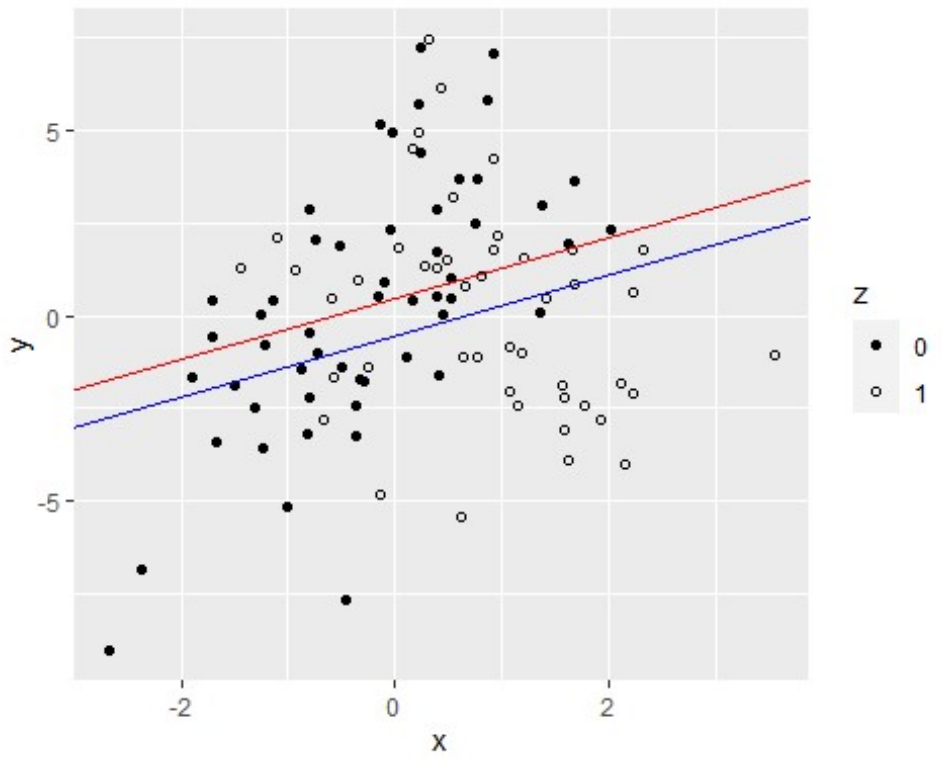
```
library(ggplot2)
simdata$z <- as.factor(simdata$z)
ggplot(simdata, aes(x=x,y=y, shape=z)) + geom_point() +
scale_shape_manual(values = c(19, 1))
```



To fit a model to this data, we can start with a model where we consider the possibility that the two groups defined by z have different intercepts, but not different slopes. That is to say that we do not include interaction variables, but we do include indicators for intercept constants.

```
library(rstanarm)

## Loading required package: Rcpp

## This is rstanarm version 2.21.1

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default
priors!

## - Default priors may change, so it's safest to specify priors, even if
equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend
calling

##   options(mc.cores = parallel::detectCores())

# Here we fit our model
fit1 <- stan_glm(y~x+z, data=simdata, refresh=0)
cf1 = coef(fit1)
# Here we display our model graphically
ggplot(simdata, aes(x=x,y=y, shape=z)) + geom_point() +
```

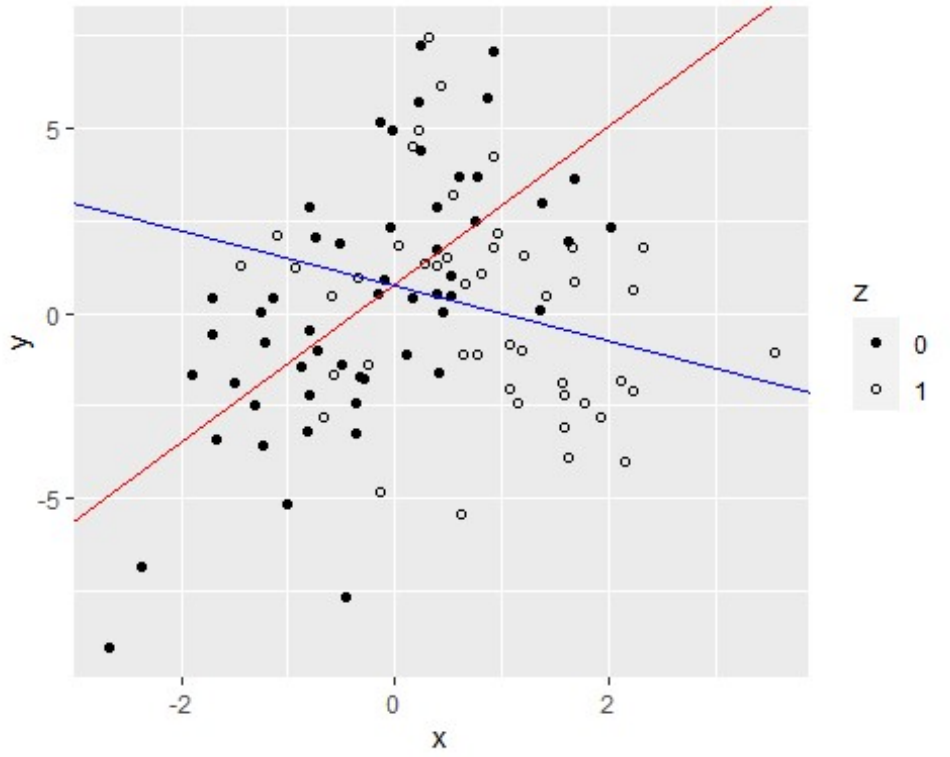
```
scale_shape_manual(values = c(19, 1)) + geom_abline(slope = cf1[2], intercept
= c(cf1[1],cf1[1]+cf1[3]), color= c("red", "blue"))
```



Note that the two regression lines above are parallel because they necessarily have the same slope, but their intercepts differ due to taking into consideration indicator variables for groups 0 and 1 (where the groups are defined by the variable z).

Now let's make some fitted lines that have interaction variables as well, meaning the model leaves room for different slopes in the different categories in addition to different intercepts.

```
# Here we fit our model
fit2 <- stan_glm(y~x*z, data=simdata, refresh=0)
cf2 = coef(fit2)
# Here we display our model graphically
ggplot(simdata, aes(x=x,y=y, shape=z)) + geom_point() +
scale_shape_manual(values = c(19, 1)) + geom_abline(slope =
c(cf2[2],cf2[2]+cf2[4]), intercept = c(cf2[1],cf2[1]+cf2[3]), color= c("red",
"blue"))
```



Note that the two lines above have different slopes as well as different intercepts and that is because we have included both indicator variables for each group as well as interaction variables that can change the slope depending on the z group the data is in.

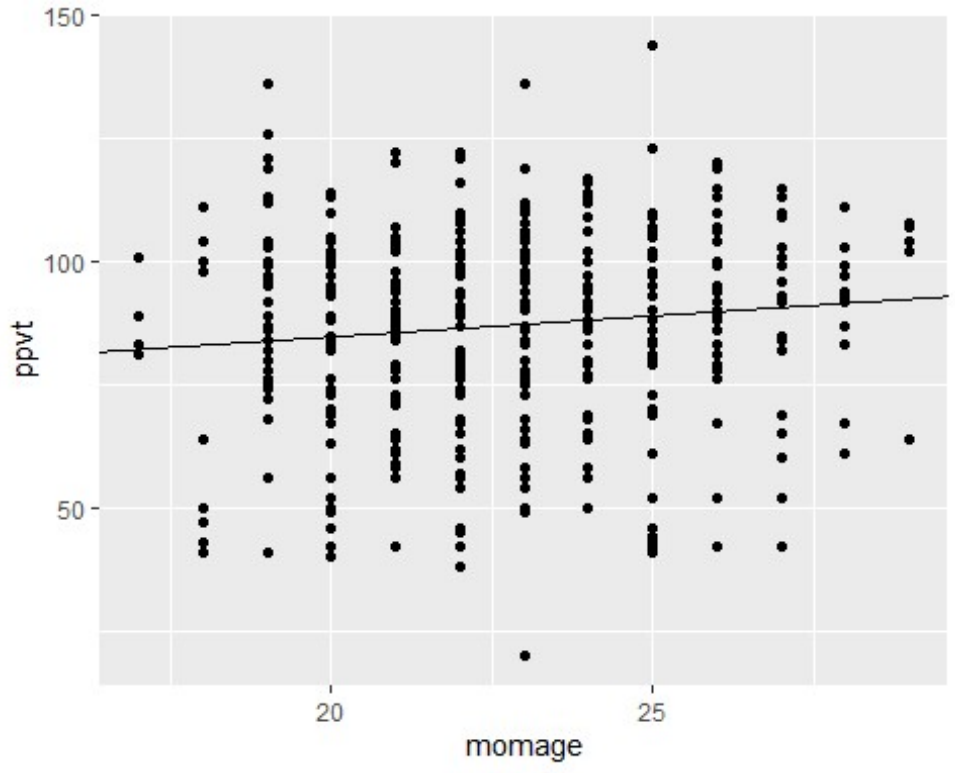
10.5: Regression Modeling and Prediction

First we can write some code to retrieve the data this problem uses and convert it into a format that we can use in R:

```
myfile <- "https://raw.githubusercontent.com/avehtari/ROS-
Examples/master/KidIQ/data/child_iq.csv"
child_iq <- read.csv(myfile, header = T)
```

Now let's fit a linear regression to the plot of the child's test scores compared with the mother's age. We can use stan_glm so that we take a Bayesian approach to fitting this regression.

```
childfit1 = stan_glm(ppvt~momage, data=child_iq, refresh=0)
cfc1 = coef(childfit1)
ggplot(child_iq, aes(x=momage,y=ppvt)) + geom_point() + geom_abline(slope =
cfc1[2], intercept = cfc1[1])
```




```
print(cfc1)

## (Intercept)      momage
##  67.5982187   0.8505964
```

The slope of the regression line fit to the above data is around 0.8 and the intercept is around 68 (note this changes every time because of the stochastic processes built into the Bayesian methods of stan_glm). The primary assumption we make when fitting this line is that the age of a mother and the score their child gets on an IQ test share a roughly linear relationship.

In our analysis, there isn't much of a reason to look at the intercept because data about child IQ of kids with moms who had them at age 0 has no relation to the real world. Based on the slope of this regression line alone, we would conclude that there is positive linear relationship between the age of a mother and the IQ score of their child. Specifically, for every year older a mom is, the child's IQ score is 0.8 points higher on average.

This would suggest that if a mother wanted to have a child with a high IQ score, they should wait as long as possible to have the kid. If we take this regression line as absolute truth, we could say that this relationship holds for at least the range of this data. The maximum age for a mom in this data is 29, so we would recommend mothers should give birth at 29.

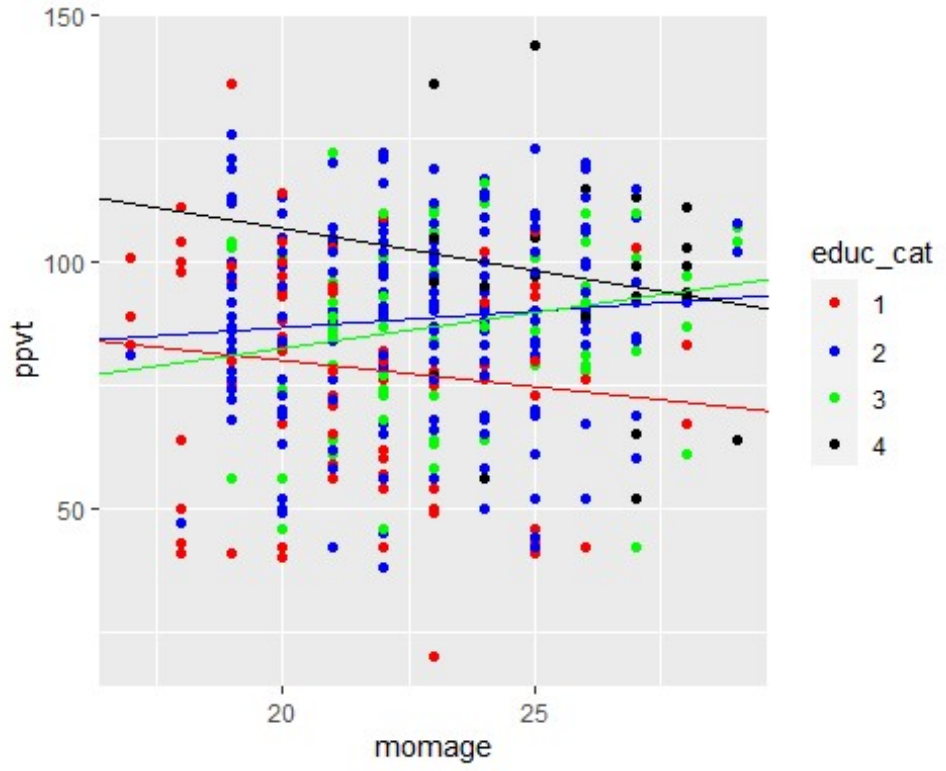
In this recommendation, there is the assumption that mother's age and child's IQ score have a linear relationship and that this linear relationship only holds for the range of this data. It is important to note that these assumptions are not necessarily reasonable to make.

Now we can repeat this regression analysis, but this time including mother's education in our analysis. We can generate such a regression with the following R code:

```
# Turn education category into a categorical variable from a numeric variable
child_iq$educ_cat <- as.factor(child_iq$educ_cat)
# Now we fit our model with both age and education as predictors
childfit2 = stan_glm(ppvt~momage + educ_cat + educ_cat:momage, data=child_iq,
refresh=0)
```

We can then graph this linear models for each education category with the data points using the following R code:

```
cf2 = coef(childfit2)
ggplot(child_iq, aes(x=momage,y=ppvt, color=educ_cat)) +
scale_color_manual(values = c("red", "blue", "green", "black")) +
geom_point() + geom_abline(slope =
c(cf2[2],cf2[2]+cf2[6],cf2[2]+cf2[7],cf2[2]+cf2[8]), intercept =
c(cf2[1],cf2[1]+cf2[3],cf2[1]+cf2[4],cf2[1]+cf2[5]), color= c("red", "blue",
"green", "black"))
```




```
sprintf("Education level 1 has slope %f", cf2[2])

## [1] "Education level 1 has slope -1.071659"

sprintf("Education level 2 has slope %f", (cf2[2]+cf2[6]))

## [1] "Education level 2 has slope 0.678903"
```

```
sprintf("Education level 3 has slope %f", (cf2[2]+cf2[7]))
```

```
## [1] "Education level 3 has slope 1.446461"
```

```
sprintf("Education level 4 has slope %f", (cf2[2]+cf2[8]))
```

```
## [1] "Education level 4 has slope -1.691587"
```

In this new regression model, we have generate 4 separate lines for the 4 separate education categories defined in the data. We can interpret the slope of these lines as we did above (once again the intercept isn't very relevant here as there are no mother's with age 0). Factoring education level into our linear models of mother's age against child's IQ score changes the conclusions we can draw from the data and the "recommendations" we would make to potential mothers.

Now we can say that at levels of education 2 and 3, the regression model generated has a positive slope which implies a positive linear relationship between mothers' age and child's IQ scores. With these two categories, we would conclude that the older the mom, the higher the child's IQ (for ages within this data range). This means we would make the same recommendation of having a child at 29 that we made above.

For education levels 1 and 4, we see that the linear relationship produced by the regression model has a negative slope which implies that there is a negative relationship between the two variables and younger mothers have higher IQ kids. Therefore, for these two groups, we would recommend having a child as young as possible (within the data range), so at age 17 [even though this seems to be a pretty extreme recommendation].

Now we are asked to create a binary indicator variable to add to this data set that indicates whether or not the mother in question graduated high school. Inferring from the comparison of the two of the data sets provided on the authors' GitHub page, I figured out that if the educational category is 2 or higher, then the mother has graduated highschool and the indicator should be 1, otherwise it should be 0.

The following code automates the process of adding this binary indicator to the data frame we are working with:

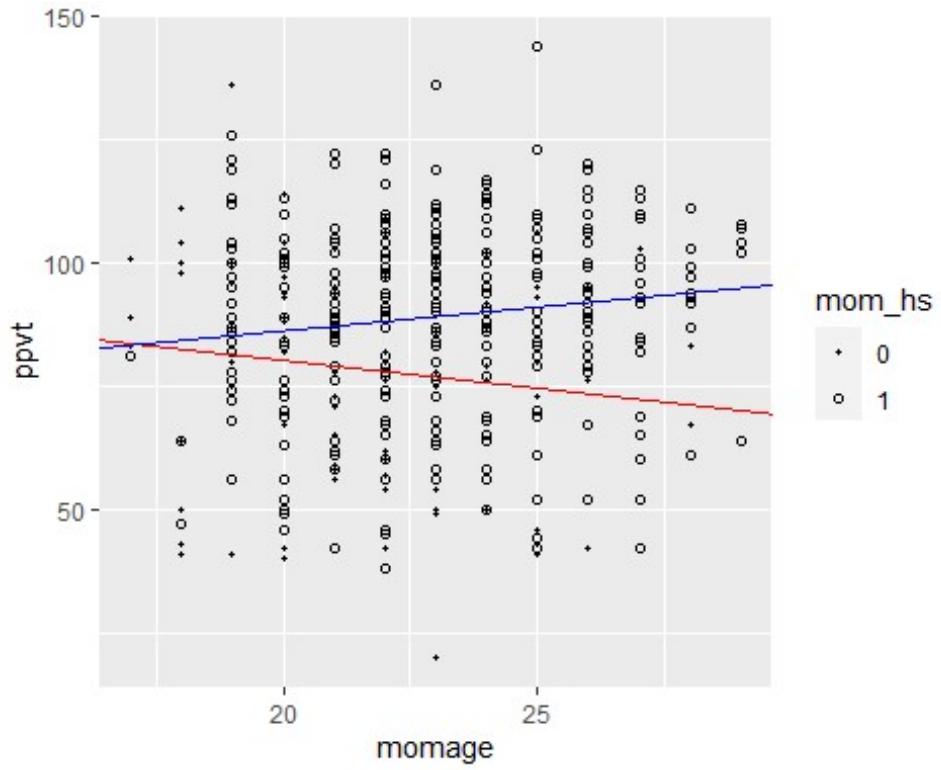
```
child_iq$mom_hs <- ifelse((child_iq$educ_cat == 1), 0, 1)
```

We can then produce a regression model that takes our newly generated binary indicator into account when creating its linear combinations. This can be done using the following R code:

```
# Turn the high school indicator variable into a categorical variable from a
numeric variable
child_iq$mom_hs <- as.factor(child_iq$mom_hs)
# Then generate a Bayesian regression model
childfit3 = stan_glm(ppvt~momage + mom_hs + mom_hs:momage, data=child_iq,
refresh=0)
```

Now let's plot our models on a shared graph along with the data points from the given information. The regression line for the non highschool graduates will be red and the regression line for the highschool graduates will be blue. This can be done with the following code:

```
cf3 = coef(childfit3)
ggplot(child_iq, aes(x=momage,y=ppvt, shape=mom_hs)) + geom_point() +
scale_shape_manual(values = c(20, 1)) + geom_abline(slope =
c(cf3[2],cf3[2]+cf3[4]), intercept = c(cf3[1],cf3[1]+cf3[3]), color= c("red",
"blue"))
```




```
sprintf("The Non-HS line has a slope of %f", cf3[2])

## [1] "The Non-HS line has a slope of -1.129164"

sprintf("The HS line has a slope of %f", (cf3[2]+cf3[4]))

## [1] "The HS line has a slope of 0.964793"
```

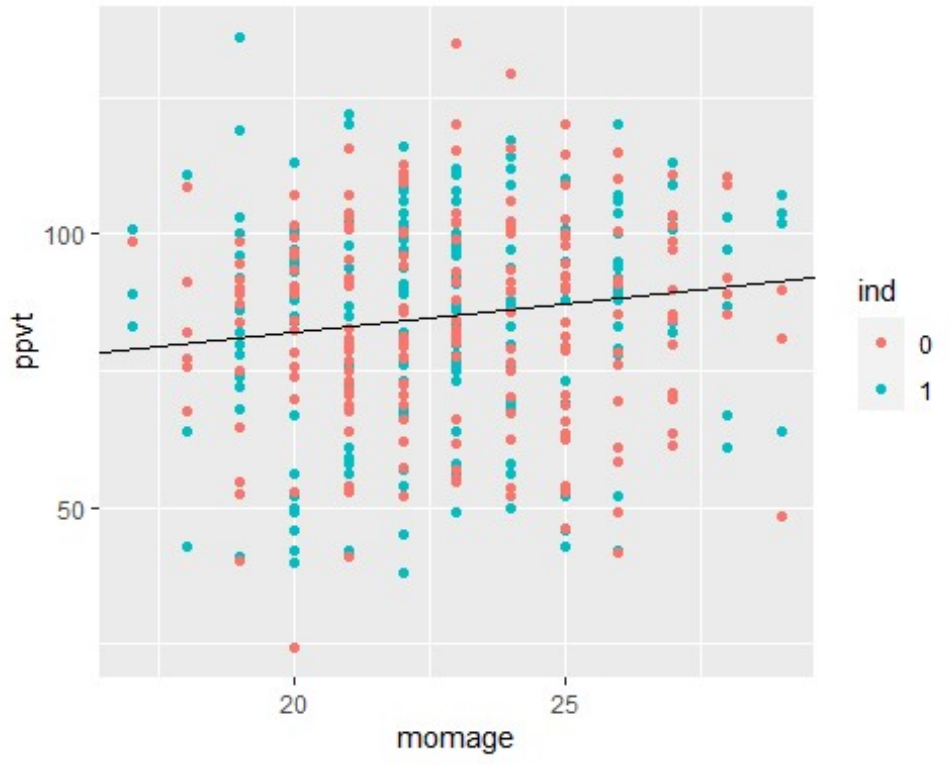
The above graph shows the differences in regression analysis results between the two groups: non-highschool graduates (0) and highschool graduates (1). Like we saw with the different educational categories, here we see that the one of the models has a positive slope (the model which corresponds to high school graduate mothers) and the other has a negative slope (the one that corresponds to non-high school graduates). This suggests that, based on the data, high school graduation of the mother has a non-trivial impact on the linear relationship between the age of a mother and the IQ of the mother's child.

Finally, the problem description asks us to generate a model fit to the first 200 data points that we will use to predict the values of the remaining 200 points in the dataset. To do this, we can use the following R code. Note that the contents of the bracket coming after child_iq specify which rows we want to base our regression model off of (here we select only the first 200 rows).

```
childfit4 = stan_glm(ppvt~momage, data=child_iq[1:200,], refresh=0)
```

Now we can use this fit to generate predictions for the last 200 points in the data set. These predictions are made by inputting the given values for mother's age in the last 200 data points into the linear model generated in childfit4 and then adding simulated error (the magnitude of this error determined using a normal distribution with standard deviation equal to the estimated value of sigma in childfit4).

```
# Getting some coefficients from the fit we generated above
cf4 = coef(childfit4)
cf4_summ = summary(childfit4)
sig = cf4_summ[3,1]
sl = cf4[2]
inter = cf4[1]
# Creating a copy of our data frame and making an indicator var for
predicted/given
chIQ_cpy = child_iq
ind = numeric(100)
ind[1:200] = 1; ind[201:400] = 0
chIQ_cpy$ind <- as.factor(ind)
# Predicting the last 200 data points and graphing the outcome
for (i in 201:400) {
  chIQ_cpy[i,1] = (chIQ_cpy[i,3] * sl) + inter + rnorm(1,0,sig)
}
ggplot(chIQ_cpy, aes(x=momage,y=ppvt, color=ind)) + geom_point() +
geom_abline(slope = cf4[2], intercept = cf4[1])
```



The above graph will be different every time this program is run because of the stochastic processes involved in simulating the error, but from what I've seen it has had results that look somewhat similar to our actual given data for this problem. This suggests that using the model generated by the first 200 points is potentially useful in predicting the outcome of the next 200 data points in this data set.

10.6: Regression Models with Interactions

First let's write some R code to retrieve the data and put it into a format we can work with in R. This is done below:

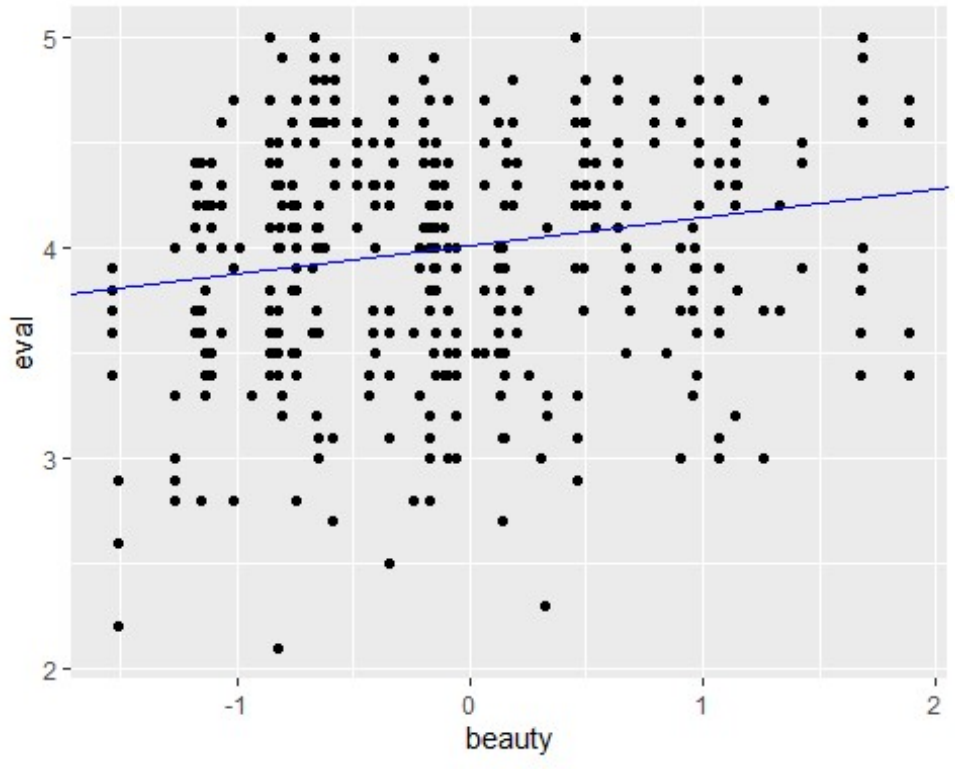
```
myfile <- "https://raw.githubusercontent.com/avehtari/ROS-
Examples/master/Beauty/data/beauty.csv"
beauty <- read.csv(myfile, header = T)
```

The first kind of regression analysis we can do is to not consider any variables other than the two we are primarily interested in: perceived beauty and evaluation score. The following R code uses Bayesian statistics to generate a regression line based on this simple model.

```
bfit = stan_glm(eval~beauty, data=beauty, refresh=0)
```

We can then graph this regression model against our data with the following R code in order to do a visual comparison as part of our analysis.

```
bc = coef(bfit)
ggplot(beauty, aes(x=beauty,y=eval)) + geom_point() + geom_abline(slope =
bc[2], intercept=bc[1], color="blue")
```



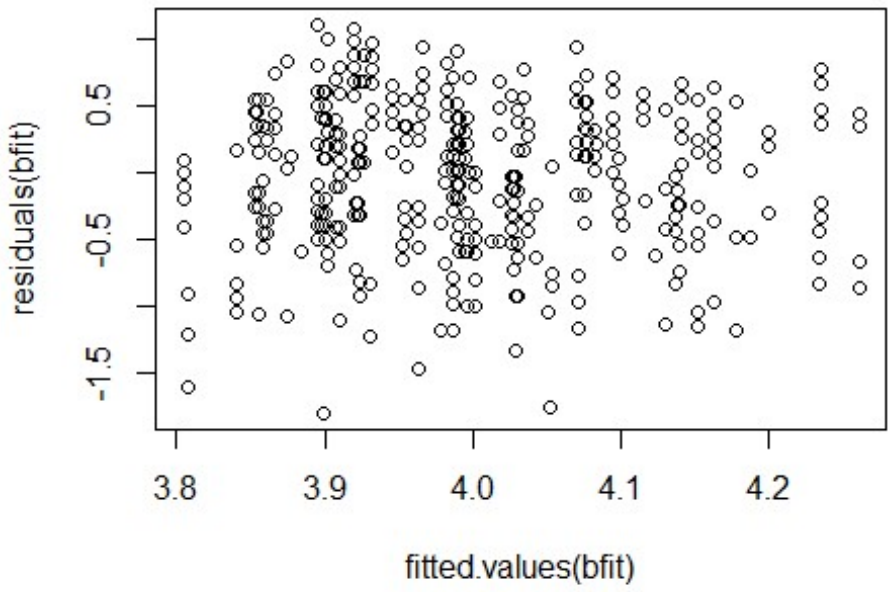

```
print(bfit)

## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty
##  observations: 463
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 4.0    0.0
## beauty      0.1    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

For the plot above, we see that the slope coefficient is about 0.13 and the intercept coefficient is about 4—although these values vary each time because of the stochastic processes involved in the backend of stan_glm. The slope coefficient suggests that there is a

slight positive linear relationship between perceived beauty and average evaluation score, such that for each point greater a teacher scores in perceived beauty, their evaluation scores increases by 0.13 on average. The intercept coefficient tells us that teachers with a 0 on this standardized beauty scale have an average evaluation score of about 4.

We can conduct some further analysis on this regression fit by plotting the residuals of this plot in the data set against the fitted values generated by our regression model. We can generate this plot using the R code provided below:

```
plot(residuals(bfit)~fitted.values(bfit))
```



The residual plot (at least in this iteration of running stan_glm) does not show any obvious patterns, which suggests that with this particular data, a linear regression model may be sufficient in describing the relationship between perceived beauty and average evaluation score of instructors.

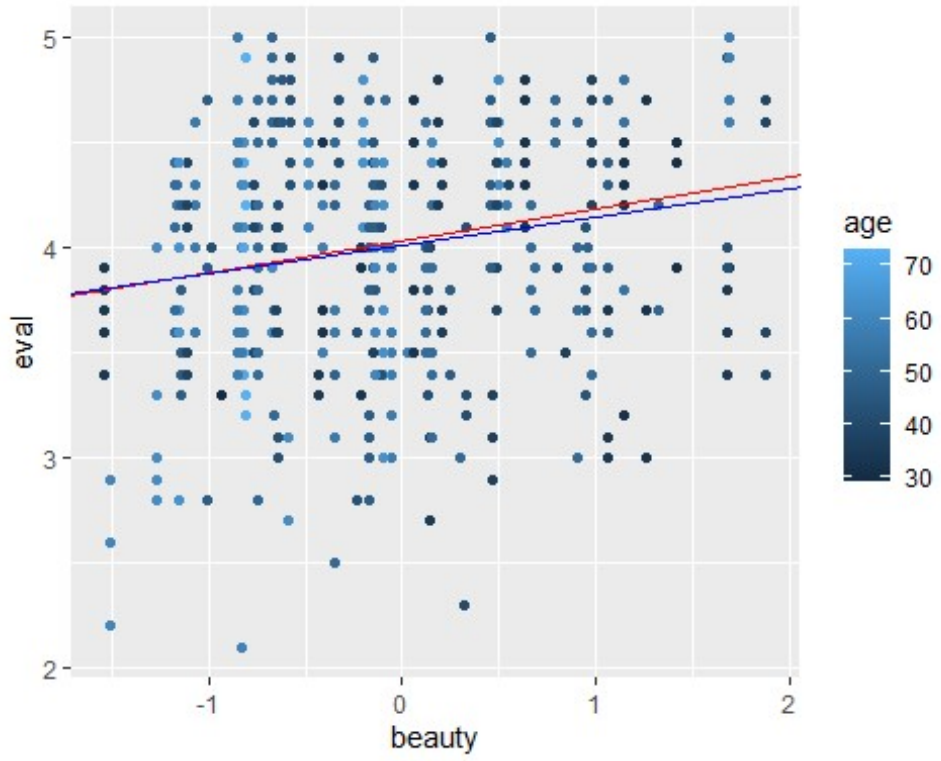
We can now factor in other variables provided in the data. Once again, we can use stan_glm to approach the regression analysis from a Bayesian perspective; but this time, we'll control for age as this is the variable provided in the data set with the most clear interaction with perceived beauty. We can control for age by including an interaction variable in our model and then setting the value for age to the average age across the dataset.

```
bfit2 = stan_glm(eval~beauty*age, data=beauty, refresh=0)
print(bfit2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty * age
##  observations: 463
##  predictors:   4
## ------
##             Median MAD_SD
## (Intercept)  4.0    0.1
## beauty      -0.3    0.1
## age          0.0    0.0
## beauty:age   0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

The above stan_glm function call generates a regression fit that includes the coefficient for the interaction variable as well as a coefficient for age as a predictor. Like discussed above, we can average age in the data and then use that as our standard age value for the linear combination. After generating this linear model, we can represent it graphically by plotting it alongside the data points given in the csv file, as well as the linear model we made previously:

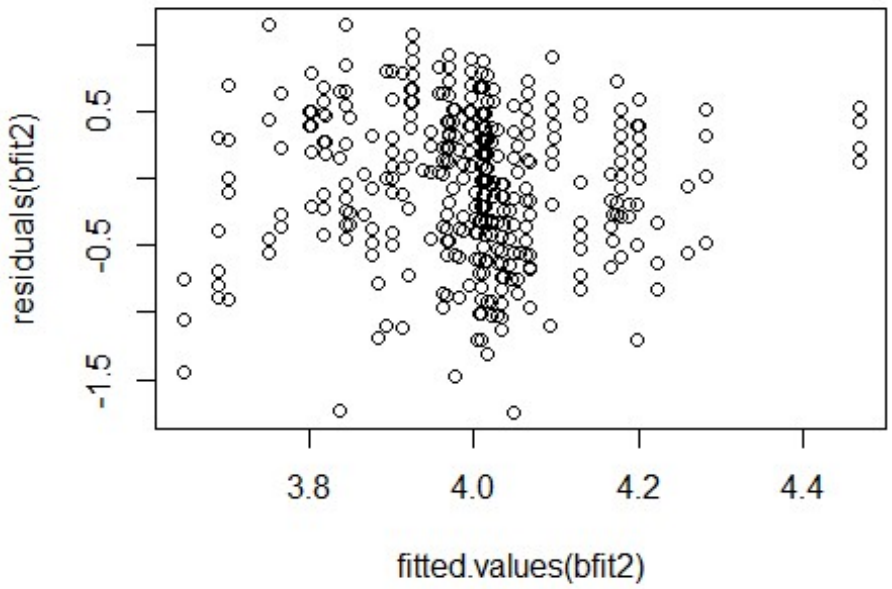
```
bc2 = coef(bfit2)
# Find average age for use in our linear combo
avg_age = mean(beauty$age)
# Calculating slope and intercept coefs with average age
inter = bc2[1] + (bc2[3]* avg_age)
sl = bc2[2] + (bc2[4] * avg_age)
# Plotting our generated regression fit
ggplot(beauty, aes(x=beauty,y=eval, color=age)) + geom_point() +
geom_abline(slope = sl, intercept = inter, color="red") + geom_abline(slope =
bc[2], intercept=bc[1], color="blue")
```



In the red line above which we generated with our new model, the slope represents the increase in evaluation score on average for every 1 point increase on the standardized perceived beauty scale among instructors that are the same age. This is slightly different than the interpretation of the slope we used earlier because we have now factored in the interactions between the age and beauty predictors.

In the graph above, we can see that these two regression models aren't too far off from each other. However, we can plot the residuals for this second model to see if there is any more noticeable difference in that regard:

```
plot(residuals(bfit2)~fitted.values(bfit2))
```



The residual plot looks different than the one we generated above, but this one also does not seem to have any clear pattern which is a good sign.

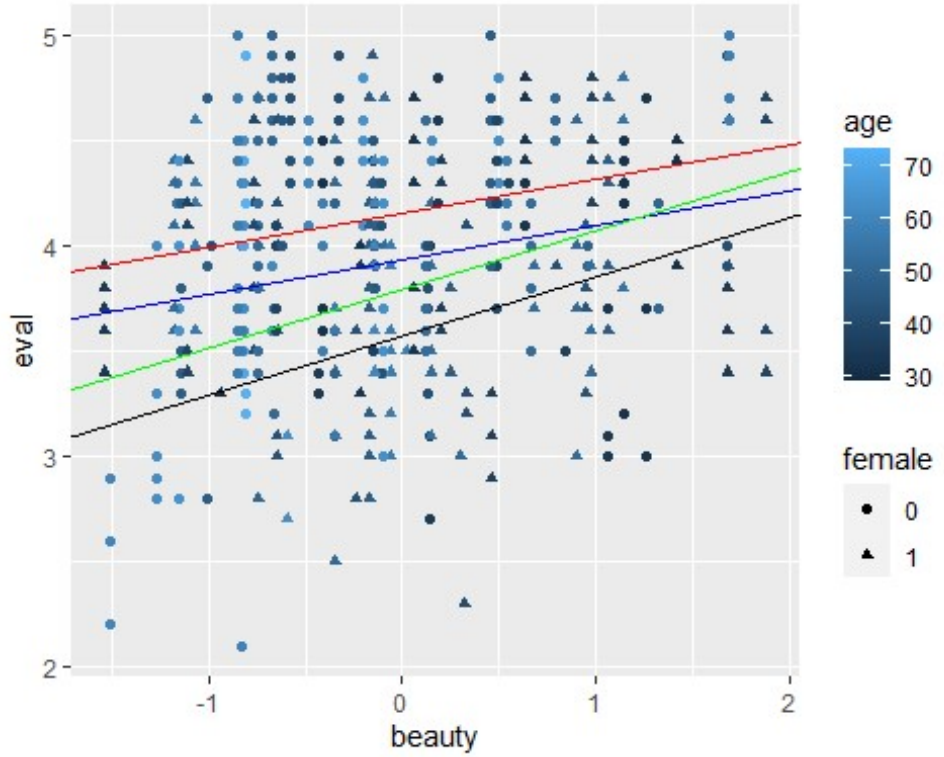
Continuing along these lines of adding in more predictors for the average evaluation score, we can make a regression model that takes into account more of the variables given to us in the csv; namely beauty, age, gender, and whether or not the instructor is a native English speaker.

```
# Change all the vairables coded in binary in our data frame to categorical
variables
# with the as.factor function
beauty$female = as.factor(beauty$female)
beauty$nonenglish = as.factor(beauty$nonenglish)
# Fit a regression line with every variable as a predictor
bfit3 = stan_glm(eval~beauty*age + beauty*female + beauty*nonenglish,
data=beauty, refresh=0)
bc3 = coef(bfit3)
print(bfit3)

## stan_glm
##  family:       gaussian [identity]
##  formula:      eval ~ beauty * age + beauty * female + beauty * nonenglish
##  observations: 463
##  predictors:   8
## ------
##                    Median MAD_SD
## (Intercept)         4.2    0.1
```

```
## beauty              -0.4    0.2
## age                  0.0    0.0
## female1             -0.2    0.0
## nonenglish1         -0.4    0.1
## beauty:age           0.0    0.0
## beauty:female1       0.0    0.1
## beauty:nonenglish1   0.1    0.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

This creates a rather hectic model with four different lines which we can plot using the R code below:

```
inter = bc3[1] + (bc3[3]* avg_age)
sl = bc3[2] + (bc3[6] * avg_age)
ggplot(beauty, aes(x=beauty,y=eval, color=age, shape=female)) + geom_point()
+ geom_abline(slope = c(sl, sl + bc3[7], sl + bc3[8], sl + bc3[7] + bc3[8]),
intercept = c(inter, inter + bc3[4], inter + bc3[5], inter + bc3[4] +
bc3[5]), color=c("red", "blue", "green", "black"))
```



In this graph, the red line shows the regression model for average evaluation against beauty for instructors who are not female and are native English speakers; the blue line is the model for instructors who are female; the green line is the model for instructors who are not native English speakers; and the black line is the model for instructors who are female and not native English speakers.

In all of these models, age is controlled for which means that the slope is supposed to represent the increase in evaluation in score on average for each point increase in beauty when comparing instructors of the same age.

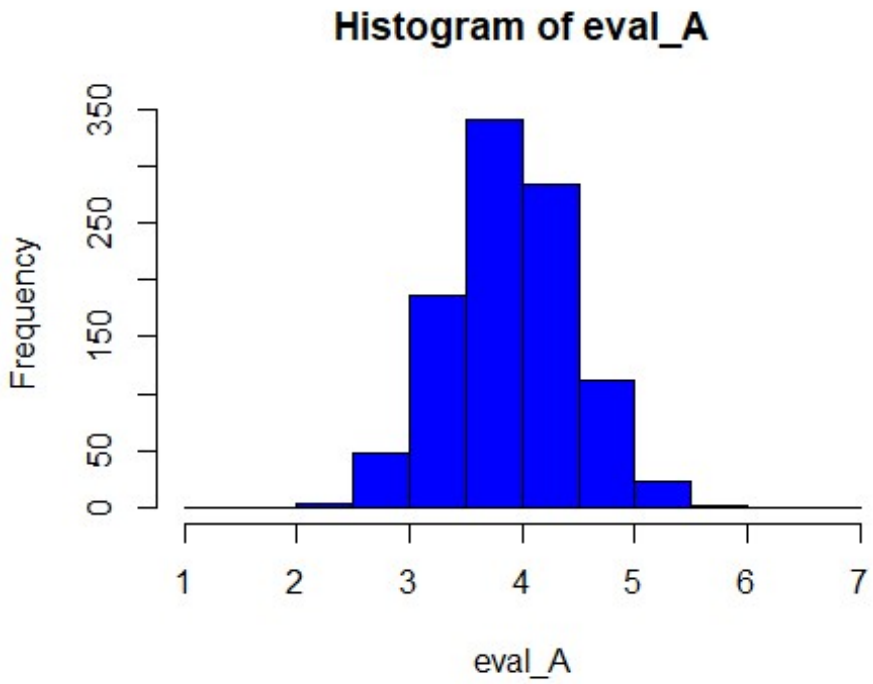
Using this model with all these interactions, we could narrow down our estimated expected values for evaluation score based on different parameters of the instructors.

10.7: Predictive simulation for linear regression

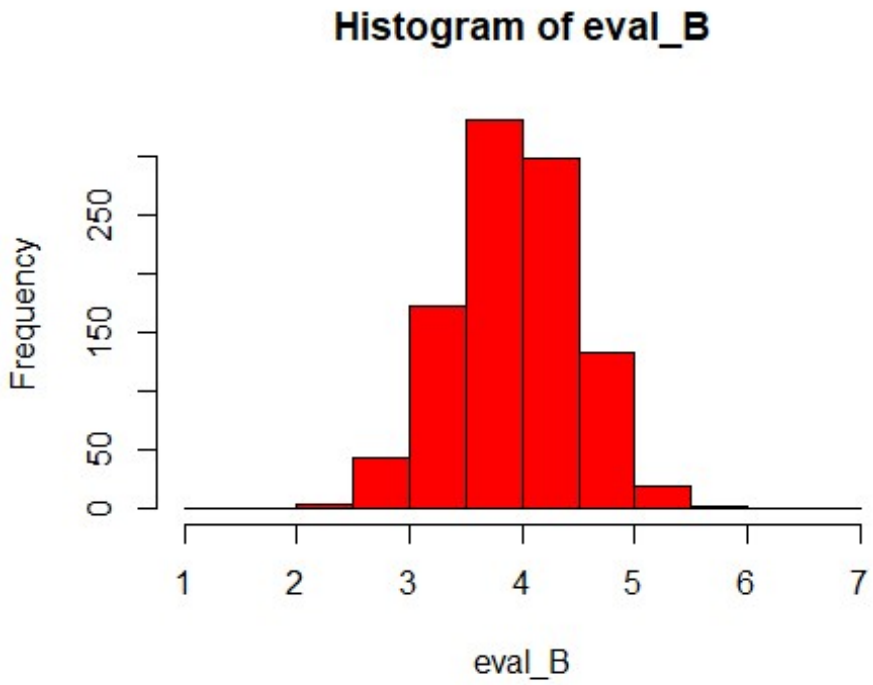
If we are considering two professors: "Instructor A, a 50-year-old woman who is a native English speaker and has a beauty score of −1" and "Instructor B, a 60-year-old man who is a native English speaker and has a beauty score of −0.5," then we can use posterior_predict along with the predictor values given in these descriptions to estimate what evaluation score they will get.

To do this, we can make two data frames that represent each of the instructors and do a simulation of 1000 draws from the posterior distribution of these instructors' evaluation scores. We can then represent these 1000 simulations in the form of histograms to compare the two visually.

```
# Making data frames representing the different instructors
instrucA = data.frame(beauty = -1, female = as.factor(1), age = 50,
nonenglish = as.factor(0))
instrucB = data.frame(beauty = -0.5, female = as.factor(0), age = 60,
nonenglish = as.factor(0))
# Using posterior_predict to simulate sampling from the posterior dist'n
eval_A = posterior_predict(bfit2, newdata = instrucA, 1000, refresh = 0)
eval_B = posterior_predict(bfit2, newdata = instrucB, 1000, refresh = 0)
# Showing the results visually
hist(eval_A, col="blue", breaks = seq(from=1, to=7, by=0.5))
```

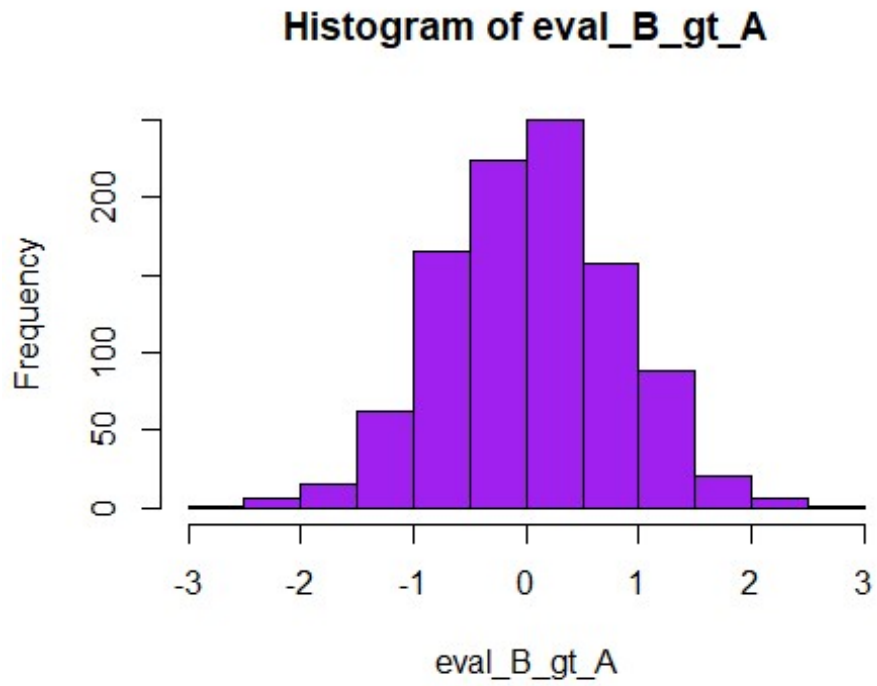



```
hist(eval_B, col="red", breaks = seq(from=1, to=7, by=0.5))
```



We can graph the simulated differences between the evaluation scores of these two professors by subtracting each simulated score of B from each simulated score of A and displaying the differences with a similar histogram visualization:

```
# Generating the simulation difference statistics
eval_B_gt_A = eval_B - eval_A
# And then displaying them with a histogram
hist(eval_B_gt_A, col="purple", breaks = seq(from=-3, to=3, by=0.5))
```



With this vector of simulated differences between evaluations of instructors B and A, we can calculate the percentage of the simulations where instructor A has a score greater than B. When this is the case, the value stored in the eval_B_gt_A vector will be negative because if A is greater than B, than the difference B-A would be negative. We can then use this calculated percentage to estimate the probability that instructor A will have an observed evaluation score greater than instructor B.

```
est_pr = length(which(eval_B_gt_A < 0)) / length(eval_B_gt_A)
sprintf("The estimated probability that instructor A has a higher evaluation
score is then %f", est_pr)

## [1] "The estimated probability that instructor A has a higher evaluation
score is then 0.475000"
```

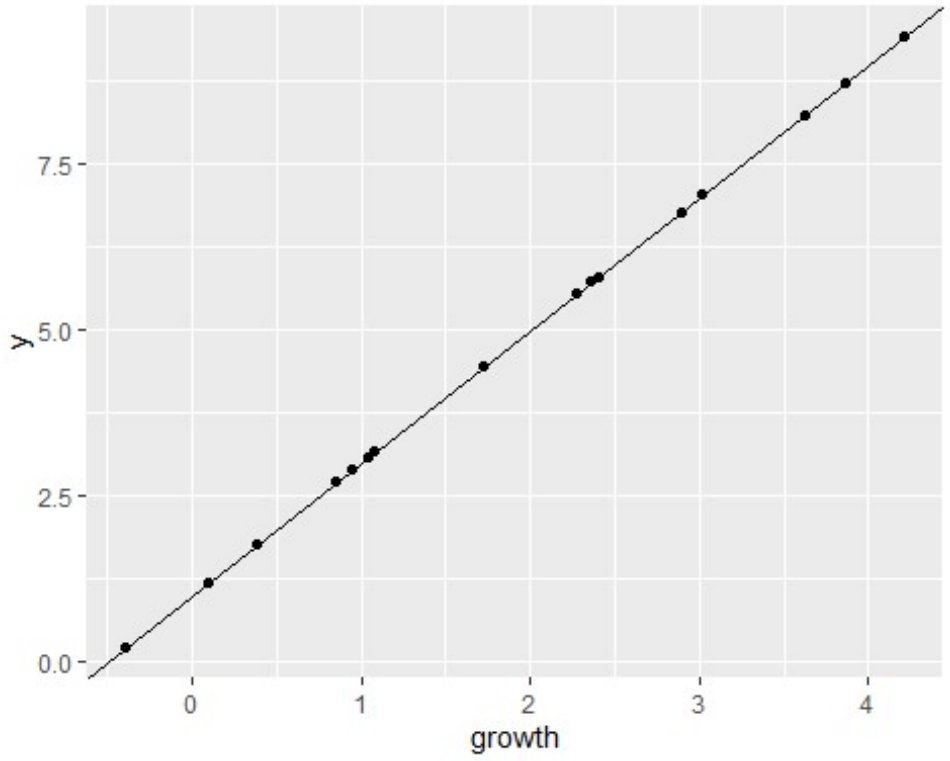
10.9: Collinearity

We can start off by importing the data this problem wants us to use and formatting it in a usable way. This can be done with the following R code:

```
myfile <- "https://raw.githubusercontent.com/avehtari/ROS-
Examples/master/ElectionsEconomy/data/hibbs.dat"
hibbs <- read.table(myfile, header = T)
```

We are now asked to make a variable that is collinear to the economic growth variable in the hibbs data set. Two variables, t and s are collinear if and only if they share some exact linear relationship t = a + b(s) where a and b are constants. Therefore, we can generate this collinear variable by producing its values according to a linear relationship with growth with arbitrary coefficients a and b. We can visually demonstrate the collinearity of these variables by displaying a graph with a regression line that goes exactly through all points.

```
# Here are our arbitrary coefficients
a = 1; b = 2
# Generating the collinear variable
y = a + (b*hibbs$growth)
hibbs$y <- y
# Fitting a regression line
reg_line = lm(y~growth, hibbs)
lc = coef(reg_line)
# Plotting the variables and the regression line
ggplot(hibbs, aes(x=growth, y=y)) + geom_point() + geom_abline(slope=lc[2],
intercept=lc[1])
```



The model we are interested in for this data set is modeling economic growth as a predictor for vote percentage won by the incumbent candidate in an election. We can add our collinear variable y into the model as a predictor, so that its values are included as a

part of the linear combination used to model the relationship between growth and vote percentage. Including y as a predictor for vote (along with the original predictor growth), we can produce a model with the following R code.

```
#growthfit = stan_glm(vote~growth*y, data=hibbs, refresh=0)
growthfit2 = lm(vote~growth*y, data=hibbs)
print(growthfit2)

##
## Call:
## lm(formula = vote ~ growth * y, data = hibbs)
##
## Coefficients:
## (Intercept)       growth            y     growth:y
##     47.2666       0.7996           NA       0.2566
```

Note that the stan_glm function call is commented out because if we try to create a regression model with 2 collinear predictors using Bayesian statistics, it results in an error that causes the stan__glm function to never terminate. The lm function terminates, but it also results in an error which we can see above in the form of the y coefficient in the fit being NA.

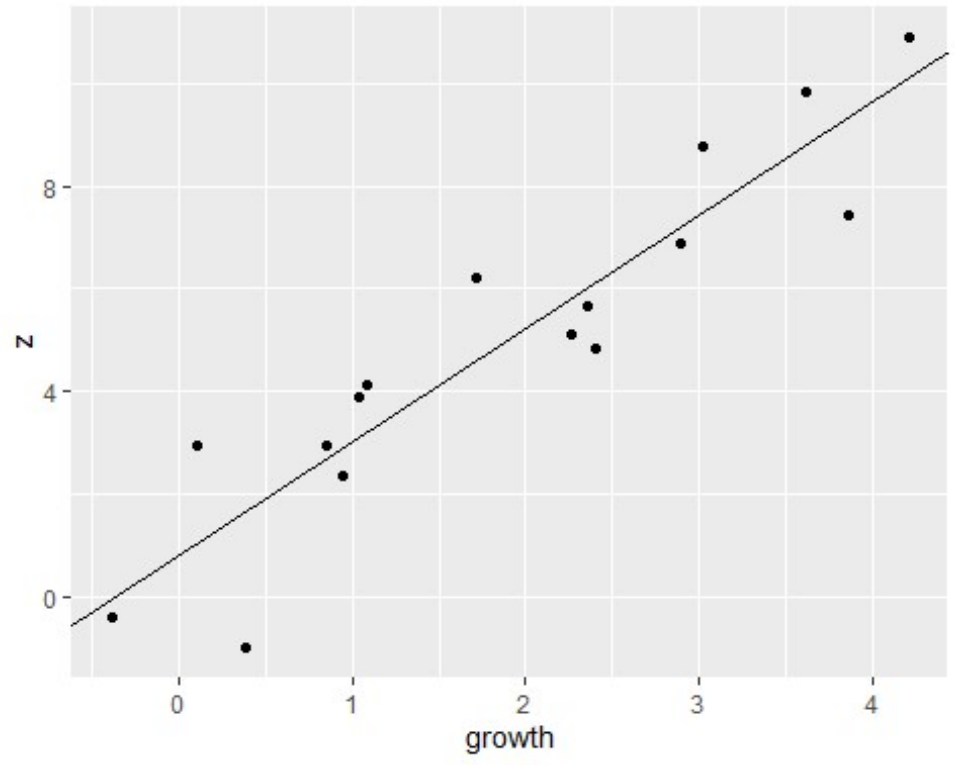
The reason both of these regression modeling functions result in errors when we input two collinear variables as predictors is that this causes the model to be nonidentifiable (i.e. the model uses parameters that cannot be estimated uniquely). If a model uses two parameters that are perfect linear combinations of each other, then their probability distributions are not sufficiently different and it may be impossible to identify the true values of the parameters (here interaction and intercept variables) even after an infinite number of observations.

We can avoid the non-identifiablity of this model by making a "nearly collinear" variable z using the same linear relationship we used to generate the variable y, but this time adding some noise that makes z not perfectly collinear with the growth variable.

```
a = 1; b = 2
# Used trial and error to come up with a sigma for our error term that came
up with a
# correlation of roughly 0.9
z= a + (b*hibbs$growth) + rnorm(16,0,1.52)
hibbs$z <- z
# Fitting a regression to the relationship of these two variables
reg_line = lm(z~growth, hibbs)
lc = coef(reg_line)
```

We can then plot this nearly collinear variable against the economic growth to show visually that they are related in a linear fashion, but do not have perfect collinearity.

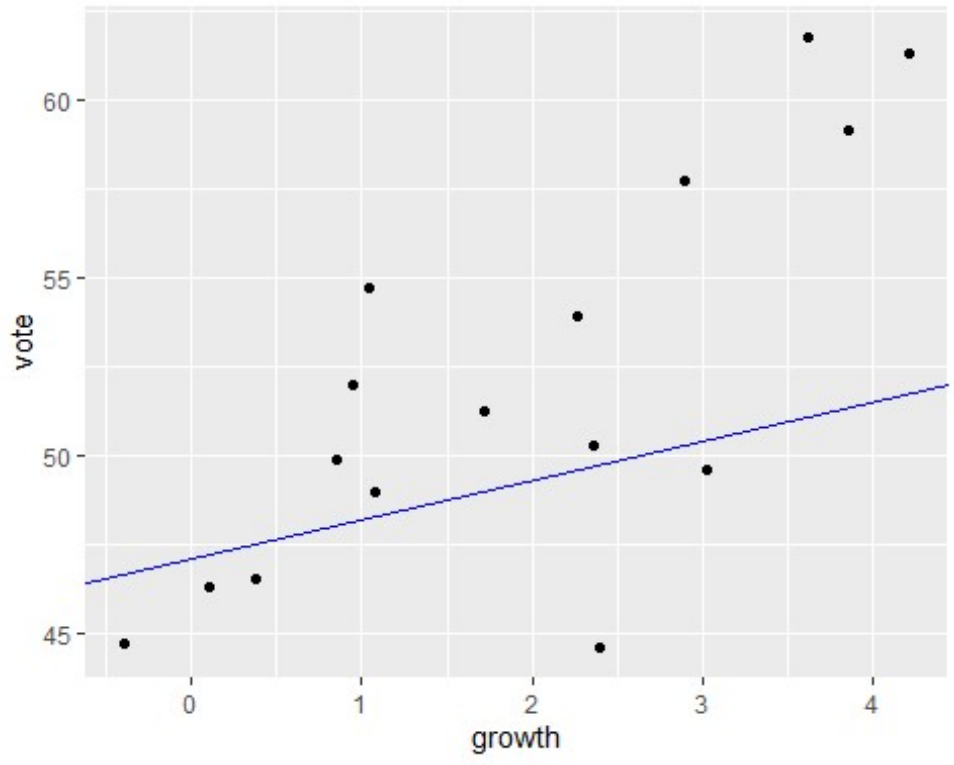
```
ggplot(hibbs, aes(x=growth, y=z)) + geom_point() + geom_abline(slope=lc[2],
intercept=lc[1])
```




```
print(cor(hibbs$growth, hibbs$z))

## [1] 0.9236434
```

Now, we can use this nearly collinear variable as a predictor in our regression model; we can produce such a model with the following R code.

```
growthfit3 = stan_glm(vote~growth*z, data=hibbs, refresh=0)
gf3 = coef(growthfit3)
ggplot(hibbs, aes(x=growth, y=vote)) + geom_point() +
geom_abline(slope=(gf3[2] + gf3[4]), intercept=(gf3[1] + gf3[3]),
color="blue")
```



Here, calling stan_glm does not produce the error that makes it run forever because the variable is no longer perfectly collinear–it is simply close to being collinear with growth. This avoids the nonidentifiablity that caused errors with the previously generated variable y. However, this doesn't ensure the regression is reliable because there are problems that come with nearly-collinear variables as well. Namely, because these variables are nearly collinear, their variation has a disproportionate impact on the regression analysis. This disproportionate impact makes the model unstable which may lead to greater uncertainty with coefficient estimates.

10.10: Regression with few data points and many predictors

We can consider other variables outside of economic growth that could predict the vote percentage that goes to the incumbent party in an election. In particular, we can look at unemployment rate, the rate of inflation, and rate of change for gdp per capita in the year of the election. We can get this data from the following online sources:

Unemployment Rate: https://raw.githubusercontent.com/avehtari/ROS-Examples/master/Unemployment/data/unemployment_simple.dat

Rate of Inflation: https://www.usinflationcalculator.com/inflation/historical-inflation-rates/

Rate of GDP per capita growth: https://fred.stlouisfed.org/series/A939RX0Q048SBEA#0

I didn't want to spend too much time data scrapping, so I manually transferred the data found in the above websites into usable formats in R below and then appended it to the hibbs data frame we have been using for this regression.

```
unemply = c(3.0, 4.1, 5.5, 5.2, 3.6, 5.6, 7.7, 7.1, 7.5, 5.5, 7.5, 5.4, 4.0,
5.5, 5.8, 8.1)
inflate = c(1.9, 1.5, 1.7, 1.3, 4.2, 3.2, 5.8, 13.5, 4.3, 4.1, 3.0, 3.0, 3.4,
2.7, 3.8, 2.1)
gdp_pc = c(2.3, 0.3, 0.5, 4.3, 3.9, 4.1, 4.4, -1.4, 6.3, 3.2, 2.2, 2.6, 2.9,
2.9, -0.8, 1.5)
hibbs$unemply <- unemply
hibbs$infl <- inflate
hibbs$gdp <- gdp_pc
```

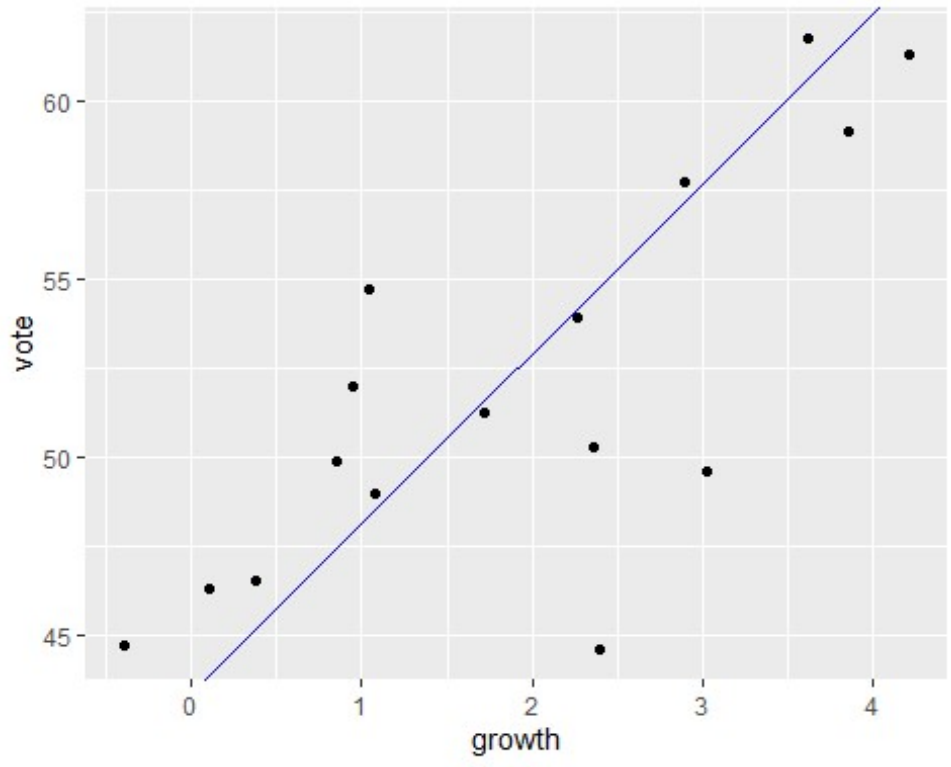
Now that we have this extra data in our data frame, we can build a regression model that takes all of these new economic indicators as additional predictors in determining the vote percentage won by the incumbent party. The R code below generates a linear regression model with all 4 predictors included in the form of both intercept indicator variables and slope interaction variables.

```
econ_fit = stan_glm(vote~growth*unemply + growth*infl + growth*gdp,
data=hibbs, refresh=0)
print(econ_fit)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      vote ~ growth * unemply + growth * infl + growth * gdp
##  observations: 16
##  predictors:   8
## ------
##                Median MAD_SD
## (Intercept)    43.5    9.0
## growth          1.1    3.5
## unemply         0.2    1.5
## infl            0.1    0.5
## gdp            -0.4    1.0
## growth:unemply  1.0    0.7
## growth:infl    -0.3    0.4
## growth:gdp     -0.4    0.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 3.3    0.8
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Now we can average the values for each predictor in order to come up with constants we can use in place of the variables to keep this graph two-dimensional and graphable. This process is done by the R code below:

```
# Calculating the means we'll need
une_avg = mean(hibbs$unemply)
inf_avg = mean(hibbs$infl)
gdp_avg = mean(hibbs$gdp)
# Calculating the slope and intercept of our regression line
ec = coef(econ_fit)
sl = ec[2] + (ec[6] * une_avg) + (ec[7] * inf_avg) + (ec[8] * gdp_avg)
inter = ec[1] + ec[3] + ec[4] + ec[5]
ggplot(hibbs, aes(x=growth, y=vote)) + geom_point() + geom_abline(slope = sl,
intercept = inter, color="blue")
```



The problem with using so many predictors with so little data points is that each data point within each predictor has the potential to have a disproportionate effect on the regression model if it is an outlier. For instance, the 13.5% inflation in 1980, as a single data point, could seriously sway the average we use for inflation which in turn could have unintended consequences on the accuracy of the model.

Furthermore, if we run into the more extreme case of having more predictors than observations, we find that the matrix representation of OLS regression breaks down as we cannot find the inverse of X^{T}X. Here, we have more observations than predictors and we are using Bayesian regression, so we do not run into this problem, but it is a possibility.