Lab 12: Regression

Name here

For this question, we will use a historical data set with NCAA tournament results.

```
library(rjags)
```

```
## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs
```

```
library(knitr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2
## --

## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
ncaa <- read_csv('https://raw.githubusercontent.com/stat456/labs/main/Lab12_data.csv') %>%
  filter(Seed.Diff != 0) %>%
  mutate(Seed.Diff = -1 * Seed.Diff,
         SAG.Diff = -1 * SAG.Diff)
```

```
## Rows: 1248 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Result
## dbl (6): Season, Score.Diff, Seed.Diff, Higher.Seed, SAG.Diff, Higher.SAG
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. (4 points) Create two figures to explore the impact of `Seed.Diff` and `SAG.Diff` on `Score.Diff`. Add a smoother line to approximate the relationship.

```
ncaa %>% ggplot(aes(y = Score.Diff, x = Seed.Diff)) +
  geom_point(alpha = .2) + theme_bw() +
  geom_smooth(method = 'loess', formula = 'y ~ x') +
  ggtitle('Point differential vs. Seed difference from historical NCAA basketball')
```

```
ncaa %>% ggplot(aes(y = Score.Diff, x = SAG.Diff)) +
  geom_point(alpha = .2) + theme_bw() +
```

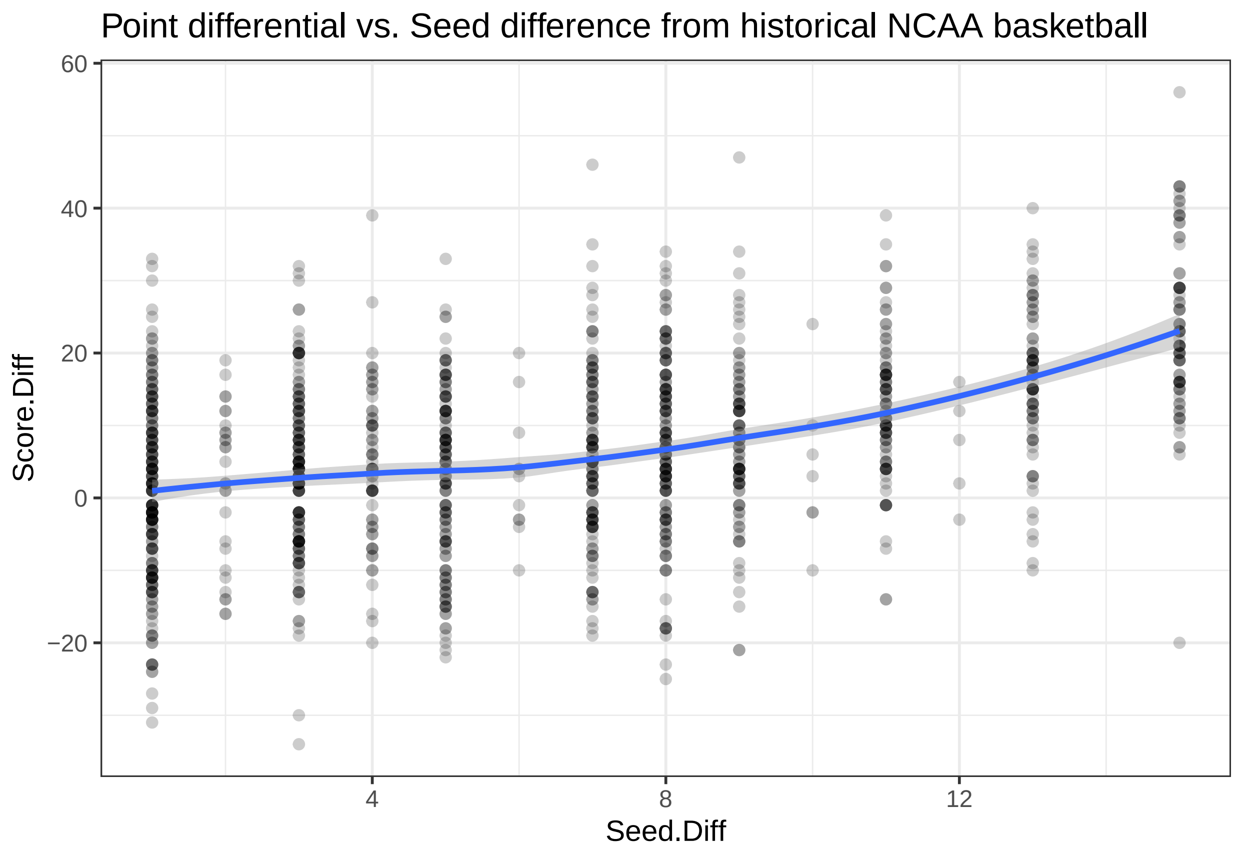

Figure 1: The higher seed would, unsurprisingly, be expected to win. We do see some potential evidence of non-linearity.

```
  geom_smooth(method = 'loess', formula = 'y ~ x') +
  ggtitle('Point differential vs. Sagarin difference from historical NCAA basketball')
```

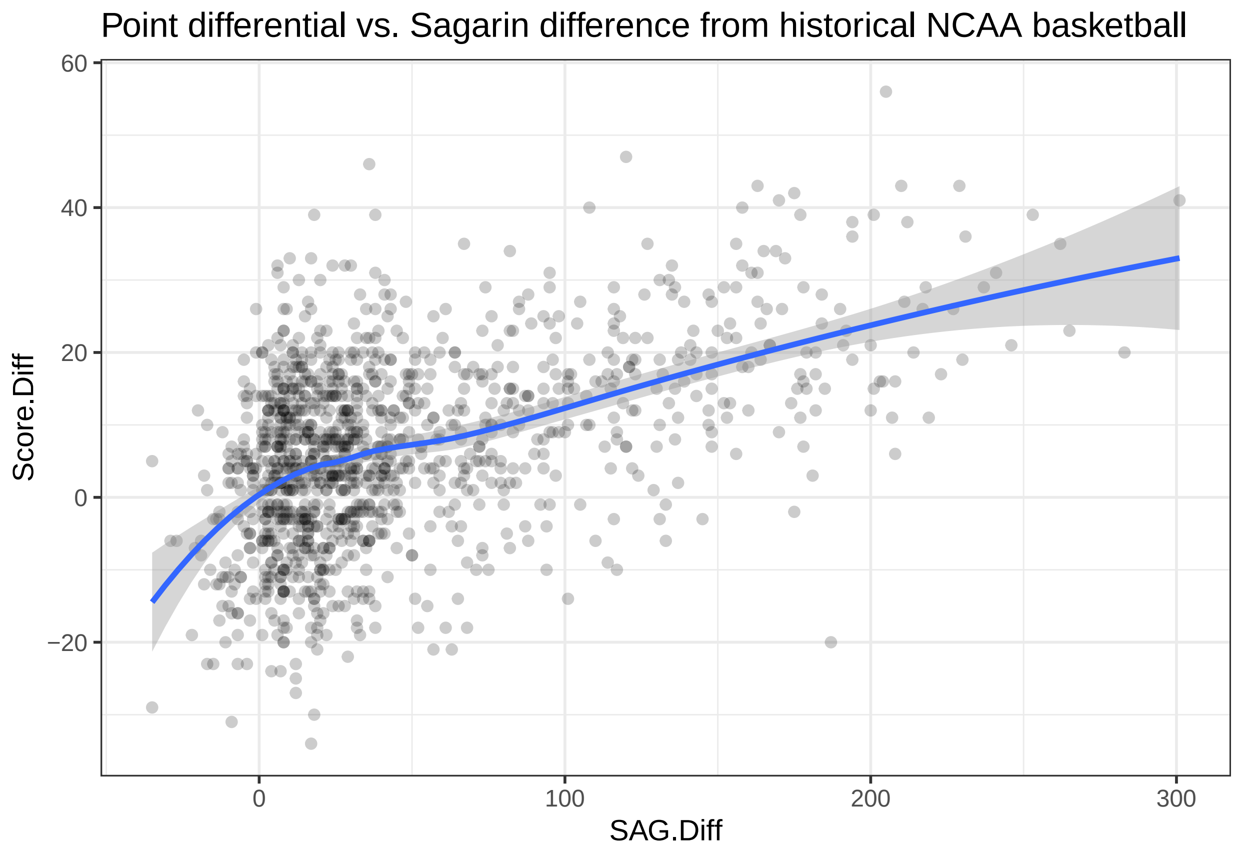

Figure 2: The negative values correspond to lower seeded teams having higher Sagarin ratings. In general the Sagarin ratings seems to do a good job of estimating score differential - even when ratings do not agree with seeds.

2. (4 points) Write a short caption to accompany each figure.

Added above

3. (4 points) Deviance Information Criteria (DIC) is a Bayesian analog to AIC. Consider the two model below, which do you prefer and why?

```
# Model String
modelString = "model {
  for ( i in 1:N ) {
    y[i] ~ dnorm(beta0 + beta1 * x[i], 1/sigma^2) # sampling model
  }
  beta0 ~ dnorm(M0,1/S0^2)
  beta1 ~ dnorm(M1, 1 / S1^2)
  sigma ~ dunif(0,C)
} "
writeLines( modelString, con='NORMmodel.txt')
```

```
# Runs JAGS Model: Seeds
seeds_model <- jags.model( file = "NORMmodel.txt",
                           data =  list(y = ncaa$Score.Diff,
                                        x = ncaa$Seed.Diff,
                                        N = nrow(ncaa),
                                        M0 = 0,
                                        S0 = .001,
                                        M1 = 1,
                                        S1 = 3,
                                        C = 100),
                         n.chains = 2, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1182
##    Unobserved stochastic nodes: 3
##    Total graph size: 2410
##
## Initializing model
```

```
update(seeds_model, n.iter = 1000)

seeds_coda <- coda.samples(seeds_model,
                          variable.names = c('beta0','beta1', 'sigma'),
                          n.iter = 5000)

summary(seeds_coda)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean        SD  Naive SE Time-series SE
## beta0 -2.146e-07 0.0009953 9.953e-06      9.953e-06
## beta1  1.123e+00 0.0439520 4.395e-04      4.395e-04
## sigma  1.169e+01 0.2394846 2.395e-03      3.057e-03
##
## 2. Quantiles for each variable:
##
##            2.5%        25%       50%       75%     97.5%
## beta0 -0.001936 -0.0006696 9.746e-07 6.749e-04  0.001922
## beta1  1.038470  1.0931228 1.123e+00 1.152e+00  1.207519
## sigma 11.236463 11.5298224 1.169e+01 1.185e+01 12.172887
```

```
dic_seed <- dic.samples(seeds_model, 5000)
dic_seed
```

```
## Mean deviance:  9166
## penalty 1.981
## Penalized deviance: 9168
```

```
# Runs JAGS Model: Sagarin
sag_model <- jags.model( file = "NORMmodel.txt",
                           data =  list(y = ncaa$Score.Diff,
                                        x = ncaa$SAG.Diff,
                                        N = nrow(ncaa),
                                        M0 = 0,
                                        S0 = .001,
                                        M1 = .5,
                                        S1 = 3,
                                        C = 100),
                         n.chains = 2, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1182
##    Unobserved stochastic nodes: 3
##    Total graph size: 2826
##
## Initializing model
```

```
update(sag_model, n.iter = 1000)

sag_coda <- coda.samples(sag_model,
                          variable.names = c('beta0','beta1', 'sigma'),
                          n.iter = 5000)

summary(sag_coda)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean        SD  Naive SE Time-series SE
## beta0 9.449e-06 0.0009991 9.991e-06      9.992e-06
## beta1 1.267e-01 0.0046265 4.626e-05      4.626e-05
## sigma 1.134e+01 0.2337037 2.337e-03      2.911e-03
##
## 2. Quantiles for each variable:
##
##            2.5%       25%       50%       75%     97.5%
## beta0 -0.001934 -0.000673 1.049e-05 6.879e-04  0.001941
## beta1  0.117572  0.123507 1.267e-01 1.298e-01  0.135735
## sigma 10.889360 11.171827 1.133e+01 1.150e+01 11.800768
```

```
dic_sag <- dic.samples(sag_model, 5000)
dic_sag
```

```
## Mean deviance:  9093
## penalty 1.979
## Penalized deviance: 9095
```

```
diffdic(dic_seed, dic_sag)
```

```
## Difference: 72.63037
## Sample standard error: 23.96276
```

DIC suggests that the Sagarin model is superior.

4. (4 points) Fit the best possible model using DIC as your criteria. You may want to consider interactions and/or non-linearity terms.

```
# Model String
modelString2 = "model {
  for ( i in 1:N ) {
    y[i] ~ dnorm(beta0 + beta1 * x[i] + beta2 * x2[i], 1/sigma^2) # sampling model
  }
  beta0 ~ dnorm(M0,1/S0^2)
  beta1 ~ dnorm(M1, 1 / S1^2)
  beta2 ~ dnorm(M2, 1 / S2^2)
  sigma ~ dunif(0,C)
} "
writeLines( modelString2, con='NORMmodel2.txt')

# Runs JAGS Model: Sagarin
sag2_model <- jags.model( file = "NORMmodel2.txt",
                          data =  list(y = ncaa$Score.Diff,
                                       x = ncaa$SAG.Diff,
                                       x2 = ncaa$Higher.SAG,
                                       N = nrow(ncaa),
                                       M0 = 0,
                                       S0 = .001,
                                       M1 = .5,
                                       S1 = 3,
                                       M2 = 0,
                                       S2 = 3,
                                       C = 100),
                         n.chains = 2, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1182
##    Unobserved stochastic nodes: 4
##    Total graph size: 4842
##
## Initializing model
```

```
update(sag2_model, n.iter = 1000)

sag2_coda <- coda.samples(sag2_model,
                          variable.names = c('beta0','beta1','beta2', 'sigma'),
                          n.iter = 5000)
```

```
summary(sag2_coda)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean       SD  Naive SE Time-series SE
## beta0 5.292e-07 0.001003 1.003e-05      9.974e-06
## beta1 1.265e-01 0.004917 4.917e-05      5.671e-05
## beta2 1.229e-03 0.020241 2.024e-04      2.347e-04
## sigma 1.134e+01 0.236345 2.363e-03      2.979e-03
##
## 2. Quantiles for each variable:
##
##           2.5%        25%       50%       75%     97.5%
## beta0 -0.00197 -0.0006779 4.383e-06 6.669e-04  0.001968
## beta1  0.11701  0.1231342 1.264e-01 1.298e-01  0.136197
## beta2 -0.03852 -0.0123127 1.601e-03 1.474e-02  0.041002
## sigma 10.88269 11.1784054 1.134e+01 1.150e+01 11.809260
```

```
dic_sag2 <- dic.samples(sag2_model, 5000)
dic_sag2
```

```
## Mean deviance:  9094
## penalty 2.944
## Penalized deviance: 9097
```

5. (4 points) Write out the formal model you've selected in part 4, including all priors.

$$pointdiff = \beta_0 + \beta_1 * x_{sagdiff} + \epsilon \quad \epsilon \sim N(0, \sigma^2) \tag{1}$$

$$\beta_0 \sim N(0, .001^2) \tag{2}$$

$$\beta_1 \sim N(1, 3^2) \tag{3}$$

$$\sigma \sim Unif(0, 100) \tag{4}$$

6. (4 points) Interpret your parameters in the model and summarize your findings. Assume you are telling your parents how statistics can be useful for filling out an NCAA bracket.

Given Sagarin ratings, seeds are not particularly useful. For each difference in Sagarin ratings, we'd expect to see about a .13 (.12, .14) expected point differential. So for two teams separated by 10 values in the ratings, the lower Sagarin rating team would be expected to win by about 1.3 points. However, there is quite a bit of randomness present, where actual spreads could differ by about 20 points in either direction.

7. (4 points) Construct a posterior predictive distribution to calculate the winning probability of the following games, Note your model is likely set to predict the point spread for the higher seed:

1. Montana State (Seed: 14, Sagarin: 133) vs. Kansas State (Seed: 3, Sagarin: 18)
2. Purdue (Seed: 1, Sagarin: 9) vs. Farleigh Dickinson (Seed: 16, Sagarin: 310)
3. Arizona (Seed: 2, Sagarin: 10) vs. Princeton (Seed 15:, Sagarin: 118)
4. Connecticut (Seed: 4, Sagarin: 4) vs. Iona (Seed: 13, Sagarin: 86)

```
# Model String
modelString_pp = "model {
  for ( i in 1:N ) {
    y[i] ~ dnorm(beta0 + beta1 * x[i], 1/sigma^2) # sampling model
  }
  pp_msu ~ dnorm(beta0 + beta1 * 115, 1/sigma^2)
  pp_purdue ~ dnorm(beta0 + beta1 * 301, 1/sigma^2)
  pp_arizona ~ dnorm(beta0 + beta1 * 108, 1/sigma^2)
  pp_uconn ~ dnorm(beta0 + beta1 * 82, 1/sigma^2)

  beta0 ~ dnorm(M0,1/S0^2)
  beta1 ~ dnorm(M1, 1 / S1^2)
  sigma ~ dunif(0,C)
} "
writeLines( modelString_pp, con='NORMmodel_pp.txt')
```

```
# Runs JAGS Model: Sagarin
sag_model_pp <- jags.model( file = "NORMmodel_pp.txt",
                           data =  list(y = ncaa$Score.Diff,
                                        x = ncaa$SAG.Diff,
                                        N = nrow(ncaa),
                                        M0 = 0,
                                        S0 = .001,
                                        M1 = .5,
                                        S1 = 3,
                                        C = 100),
                         n.chains = 2, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1182
##    Unobserved stochastic nodes: 7
##    Total graph size: 2834
##
## Initializing model
```

```
update(sag_model_pp, n.iter = 1000)

sag_coda_pp <- coda.samples(sag_model_pp,
                          variable.names = c('beta0','beta1', 'sigma',
                                             'pp_msu','pp_purdue','pp_arizona','pp_uconn'),
                          n.iter = 5000)

pp_out <- tibble(vals = c(sag_coda_pp[[1]][,'pp_msu'],
                          sag_coda_pp[[1]][,'pp_purdue'],
                          sag_coda_pp[[1]][,'pp_arizona'],
                          sag_coda_pp[[1]][,'pp_uconn']),
                 type = rep(c('Kansas St',
                              'Purdue',
                              'Arizona',
                              'UConn'),
                            each = 5000))

pp_out %>% group_by(type) %>%
  summarize(`upset prob` = mean(vals < 0)) %>%
  kable(digits = 3)
```

type	upset prob
Arizona	0.119
Kansas St	0.096
Purdue	0.000
UConn	0.177