Lab 13: Logistic Regression

Name here

For this question, we will use a historical data set with NCAA tournament results. However, we will now look at predicting a binary outcome, win or loss by the higher seeded team.

```
library(rjags)
library(knitr)
library(tidyverse)
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,
         upset = as.numeric(Result == 'Loss'))

ncaa_train <- ncaa %>% filter(Season < 2020)
ncaa_test <- ncaa %>% filter(Season >= 2020)
```

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

```
ncaa %>% ggplot(aes(y = upset, x = Seed.Diff)) +
  geom_jitter(alpha = .2, height = .05, width = .3) + theme_bw() +
  geom_smooth(method = 'loess', formula = 'y ~ x') +
  ggtitle('Upsets vs. Seed difference from historical NCAA basketball')
```

```
ncaa %>% ggplot(aes(y = upset, x = SAG.Diff)) +
  geom_jitter(alpha = .2, height = .05, width = .3) + theme_bw() +
  geom_smooth(method = 'loess', formula = 'y ~ x') +
  ggtitle('Upset vs. Sagarin difference from historical NCAA basketball')
```

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
logistic = "model {
  for ( i in 1:N ) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = beta0 + beta1 * x[i]
  }
  beta0 ~ dnorm(M0, 1 / S0^2)
  beta1 ~ dnorm(M1, 1 / S1^2)
```

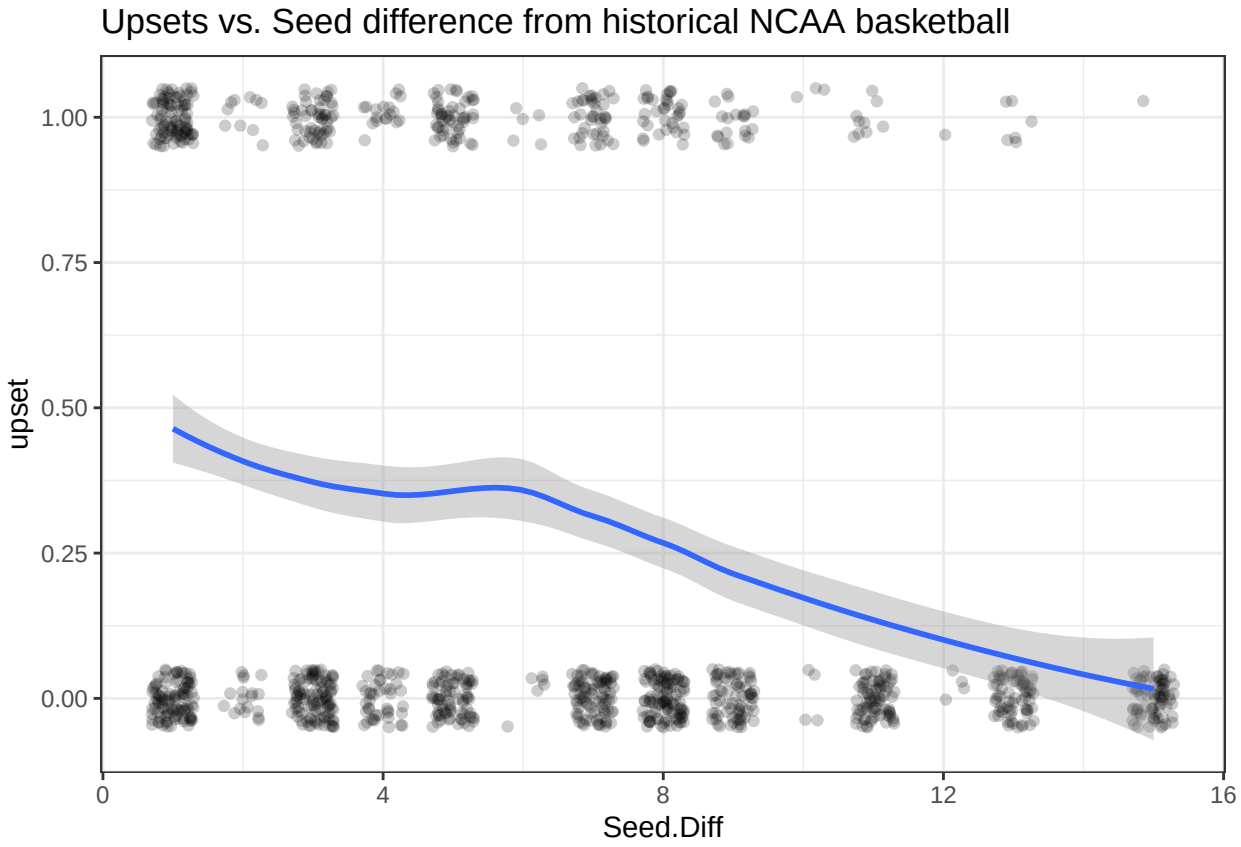

Figure 1: The higher seed would, unsurprisingly, be expected to win. We do see some potential evidence of non-linearity and non-zero upset probability.

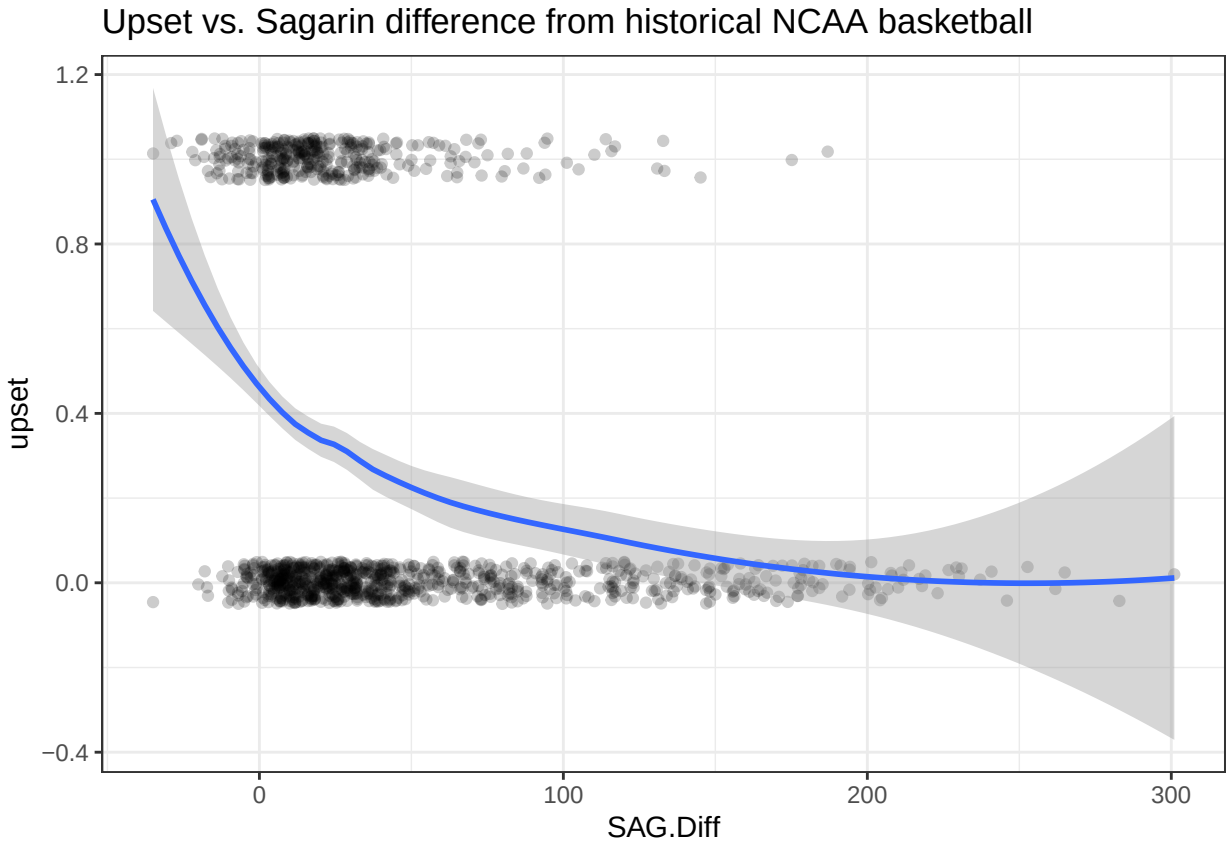

Figure 2: The negative values correspond to lower seeded teams having higher Sagarin ratings. In general the Sagarin ratings seems to do a good job of estimating upset probability - even when ratings do not agree with seeds.

```
} "
writeLines( logistic, con='logistic.txt')
```

```
# Runs JAGS Model: Seeds
seeds_model <- jags.model( file = "logistic.txt",
                           data =  list(y = ncaa_train$upset,
                                        x = ncaa_train$Seed.Diff,
                                        N = nrow(ncaa_train),
                                        M0 = 0,
                                        S0 = .001,
                                        M1 = 0,
                                        S1 = 3),
                         n.chains = 2, n.adapt = 5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1058
##    Unobserved stochastic nodes: 2
##    Total graph size: 2171
##
## Initializing model
```

```
update(seeds_model, n.iter = 5000)

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

summary(seeds_coda)
```

```
##
## Iterations = 10001:15000
## 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 -5.094e-06 0.00100 0.0000100      1.239e-05
## beta1 -1.675e-01 0.01139 0.0001139      1.489e-04
##
## 2. Quantiles for each variable:
##
##            2.5%        25%        50%        75%     97.5%
## beta0 -0.001932 -0.0006962 -1.269e-05  0.0006726  0.001968
## beta1 -0.190442 -0.1751668 -1.673e-01 -0.1597343 -0.145930
```

```
summary(glm(upset ~ Seed.Diff -1, family = binomial, data = ncaa_train))
```

```
##
## Call:
```

```
## glm(formula = upset ~ Seed.Diff - 1, family = binomial, data = ncaa_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1073  -0.8487  -0.6334   1.2492   2.2742
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Seed.Diff -0.16717    0.01136  -14.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1466.7  on 1058  degrees of freedom
## Residual deviance: 1162.7  on 1057  degrees of freedom
## AIC: 1164.7
##
## Number of Fisher Scoring iterations: 4
```

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

```
## Mean deviance:  1164
## penalty 0.9951
## Penalized deviance: 1165
```

```
# Runs JAGS Model: Sagarin
sag_model <- jags.model( file = "logistic.txt",
                         data =  list(y = ncaa_train$upset,
                                      x = ncaa_train$SAG.Diff,
                                      N = nrow(ncaa_train),
                                      M0 = 0,
                                      S0 = .001,
                                      M1 = 0,
                                      S1 = 3),
                       n.chains = 2, n.adapt = 5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1058
##    Unobserved stochastic nodes: 2
##    Total graph size: 2771
##
## Initializing model
```

```
update(sag_model, n.iter = 5000)

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

summary(sag_coda)
```

```
##
## Iterations = 10001:15000
## 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 -5.941e-06 0.0009905 9.905e-06      1.237e-05
## beta1 -2.501e-02 0.0019761 1.976e-05      2.475e-05
##
## 2. Quantiles for each variable:
##
##            2.5%        25%        50%       75%     97.5%
## beta0 -0.001965 -0.0006762  2.287e-06  0.000679  0.001909
## beta1 -0.028960 -0.0263400 -2.496e-02 -0.023687 -0.021224
```

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

```
## Mean deviance:  1146
## penalty 0.949
## Penalized deviance: 1147
```

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

```
## Difference: 17.63279
## Sample standard error: 15.22937
```

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.

omitted

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

$$upset_i \sim Bernoulli(p_i) \tag{1}$$
$$logit(p_i) = \beta_0 + \beta_1 * x_{sagdiff} \tag{2}$$
$$\beta_0 \sim N(0, .001^2) \tag{3}$$
$$\beta_1 \sim N(0, 3^2) \tag{4}$$
$$\tag{5}$$

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.

Using the Sagarin ratings, the probability of an upset is related to the difference in ratings between the two teams. Unsurprisingly, the larger the difference in ratings the lower the probability of an upset. If your goal is to fill out a bracket, then picking the team with the better Sagarin rating would be a good strategy, but still expect upsets to happen.

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
logistic_pp = "model {
  for ( i in 1:N ) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = beta0 + beta1 * x[i]
  }
  beta0 ~ dnorm(M0, 1 / S0^2)
  beta1 ~ dnorm(M1, 1 / S1^2)
  pp_msu ~ dbern(ilogit(beta0 + beta1 * 115))
  pp_purdue ~ dbern(ilogit(beta0 + beta1 * 301))
  pp_arizona ~ dbern(ilogit(beta0 + beta1 * 108))
  pp_uconn ~ dbern(ilogit(beta0 + beta1 * 82))
} "
writeLines( logistic_pp, con='logistic_pp.txt')
```

```
# Runs JAGS Model: Sagarin
sag_model_pp <- jags.model( file = "logistic_pp.txt",
                            data =  list(y = ncaa_train$upset,
                                         x = ncaa_train$SAG.Diff,
                                         N = nrow(ncaa_train),
                                         M0 = 0,
                                         S0 = .001,
                                         M1 = 0,
                                         S1 = 3),
                          n.chains = 2, n.adapt = 5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1058
##    Unobserved stochastic nodes: 6
##    Total graph size: 2779
##
## Initializing model
```

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

sag_coda_pp <- coda.samples(sag_model_pp,
                            variable.names = c('beta0','beta1',
                                               '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 == 1)) %>%
  kable(digits = 4)
```

type	upset prob
Arizona	0.0644
Kansas St	0.0580
Purdue	0.0004
UConn	0.1182

8. (1 EC point) Rather than using DIC as a model selection criteria, compare your model's ability to predict outcomes during the 2021 and 2022 tournaments (`ncca_test`).

Consider using classification error (win/loss) and Brier score (`brier.scoer()` in the `iterativeBMA` package.)