Bayes Final Project

AUTHOR
Emory Moore, Ryan Juricic

Predicting Song Popularity on Spotify

Reading in the Data and Importing Libraries

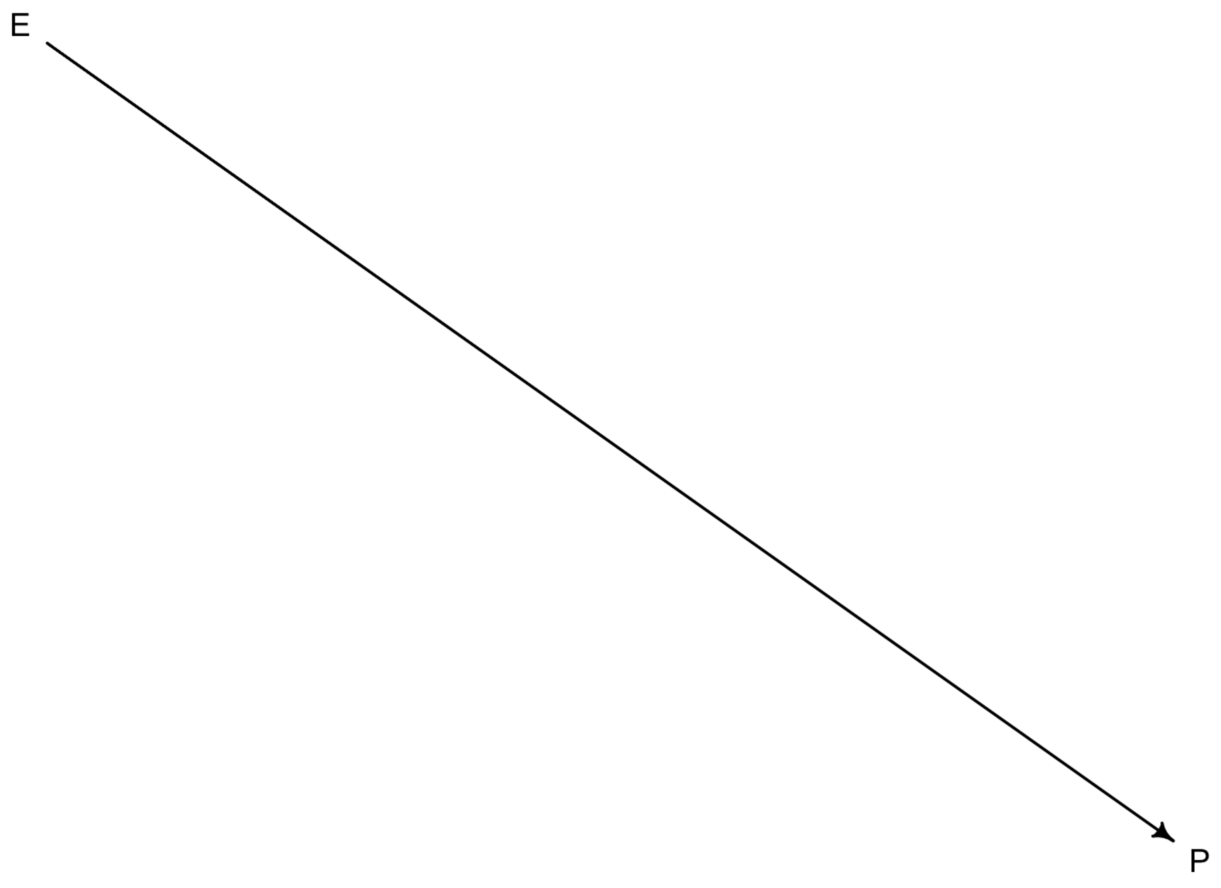
```r
library(rethinking)
library(dagitty)
library(dplyr)

data <- read.csv('/Users/ryanjuricic/Downloads/SpotifyData.csv')
d <- data

d$P <- standardize(d$popularity)
d$D <- standardize(d$danceability)
d$E <- ifelse(d$explicit == "False", 0, 1)
```

Describing the Data

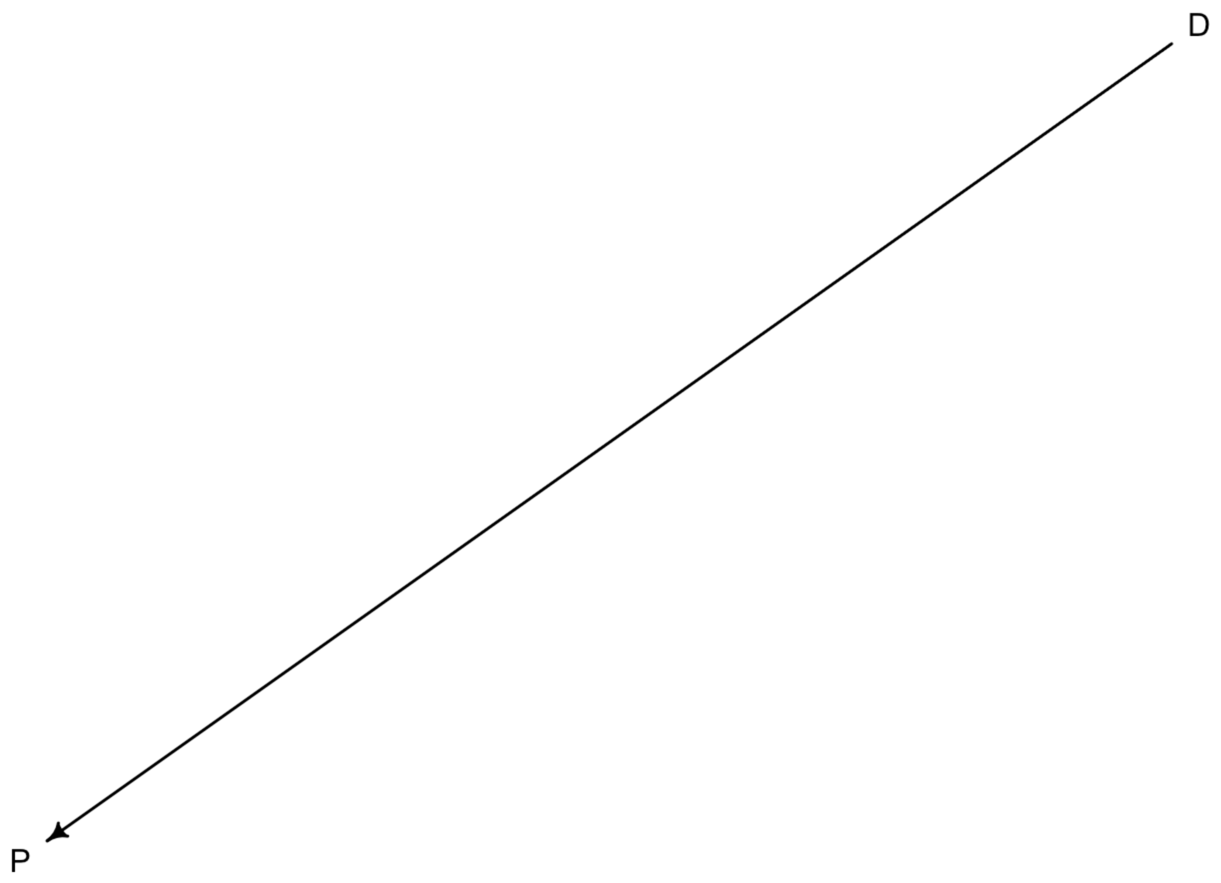
The dataset we will be using comes from Kaggle and is the Spotify Tracks Dataset. It includes 114,000 songs and several data points on each. For our model we will be trying to predict song popularity, which is rated on a scale from 0-100 with 100 being most popular. To predict popularity, we will use Explicit which a boolean variables where 0 equals not explicit and 1 equals explicit, as well as using Danceability which continuous variable from 0 to 1 with 1 being “most dance-able”.

Maharshi Pandya. (2022). 🎹 Spotify Tracks Dataset [Data set]. Kaggle. Retrieved 6 December, 2024 from https://doi.org/10.34740/KAGGLE/DSV/4372070

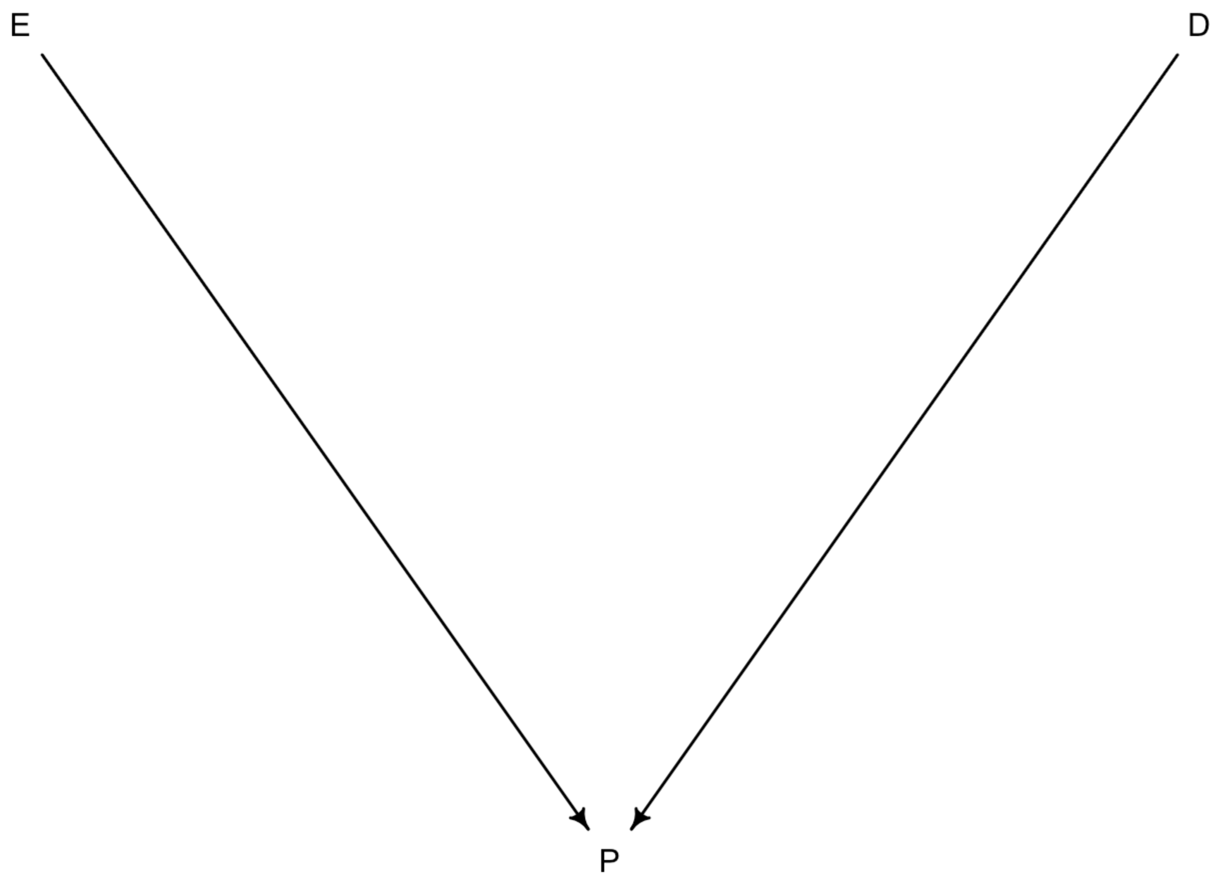
```r
# Just Explicit and Popularity
dag1 <- dagitty("dag{ E -> P }")
coordinates(dag1) <- list(x = c(E = 0, P = 1), y = c(E = 0, P = 1))
drawdag(dag1)
```


$$\begin{tikzcd}
E \arrow[rd] & \\
& P
\end{tikzcd}$$

```
# Just Danceability and Popularity
dag2 <- dagitty("dag{ D -> P }")
coordinates(dag2) <- list(x = c(D = 2, P = 1), y = c(D = 0, P = 1))
drawdag(dag2)
```



```r
# Danceability, Explicit, and Popularity
dag3 <- dagitty("dag{ D -> P; E -> P }")
coordinates(dag3) <- list(x = c(D = 2, E = 0,  P = 1), y = c(D = 0, P = 1, E = 0)
drawdag(dag3)
```



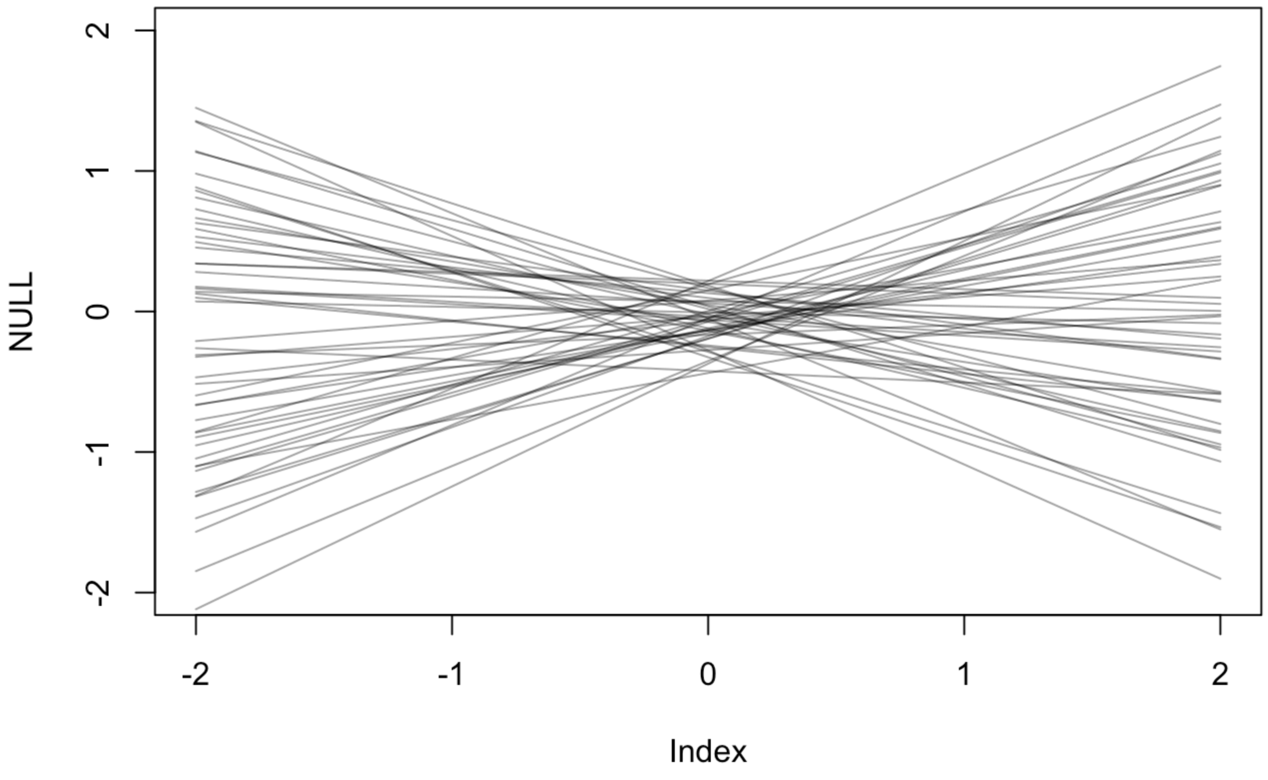
$$\begin{tikzcd}
E \arrow[dr] & & D \arrow[dl] \\
& P &
\end{tikzcd}$$

Creating Quadratic Approach Models

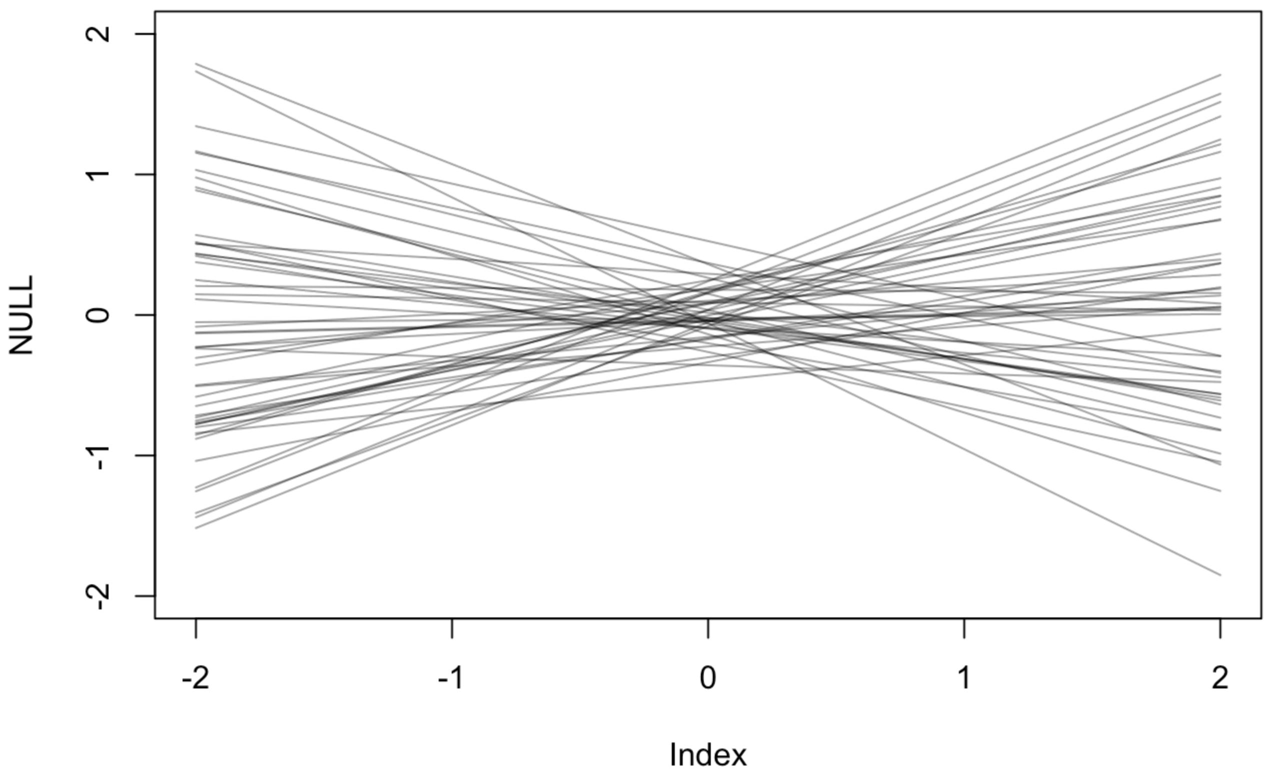
```
# Predicting Popularity with Explicit
modelQ.1 <- quap(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bE * E,
    a ~ dnorm(0, 0.2),
    bE ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ),
  data = d
)

# Predicting Popularity with Danceability
modelQ.2 <- quap(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bD * D,
    a ~ dnorm(0, 0.2),
    bD ~ dnorm(0, 0.40),
    sigma ~ dexp(1)
  ),
  data = d
)

# Predicting Popularity with Danceability and Explicit
modelQ.3 <- quap(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bD * D + bE * E,
    a ~ dnorm(0, 0.2),
    bD ~ dnorm(0, 0.4),
    bE ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ),
  data = d
)
```

Testing Prior Selection

```
set.seed(10)
prior <- extract.prior(modelQ.1)
mu <- link(modelQ.1, post = prior, data = list(E = c(-2, 2)))
plot(NULL, xlim = c(-2, 2), ylim = c(-2, 2))
for (i in 1:50) lines(c(-2, 2), mu[i, ], col = col.alpha("black", 0.4))
```



```
prior <- extract.prior(modelQ.2)
mu <- link(modelQ.2, post = prior, data = list(D = c(-2, 2)))
plot(NULL, xlim = c(-2, 2), ylim = c(-2, 2))
for (i in 1:50) lines(c(-2, 2), mu[i, ], col = col.alpha("black", 0.4))
```



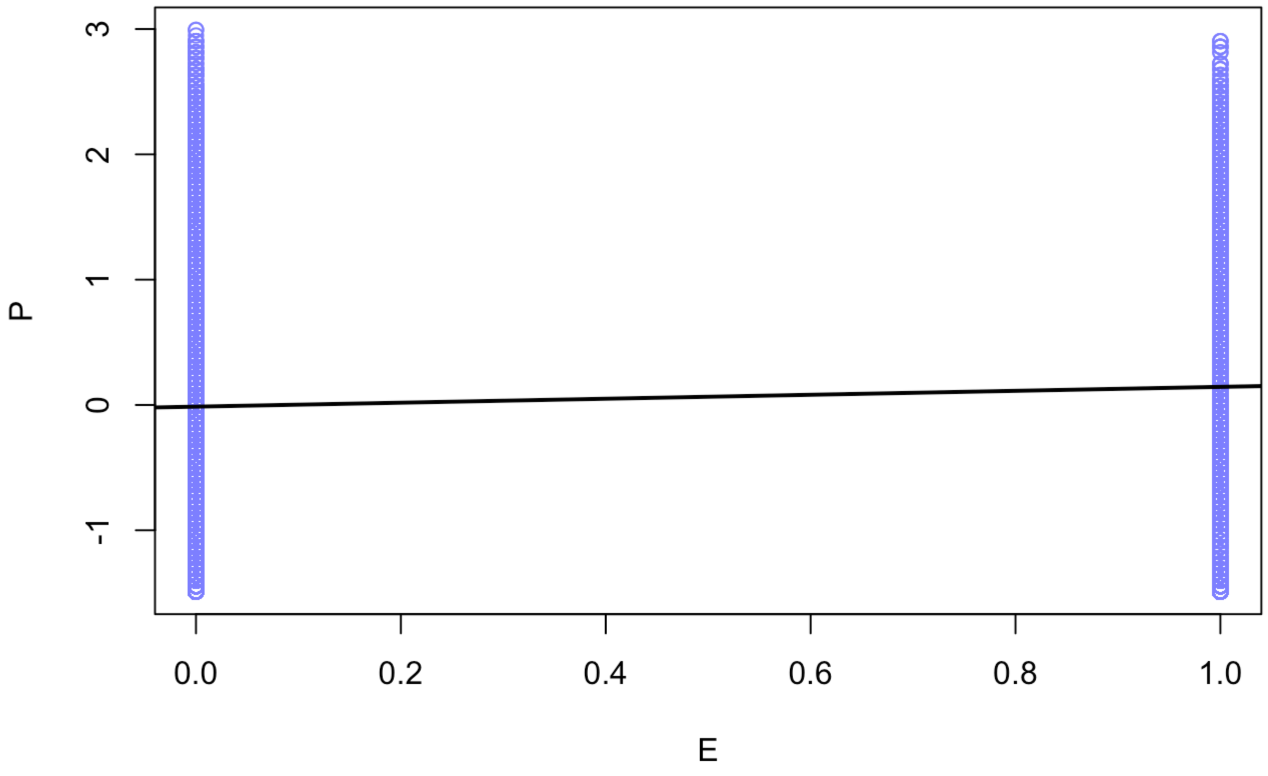
We chose mean of 0 and standard deviation of 0.5 for our Explicit priors due to the fact that Explicit is a binary classification. This gave us a good selection to work with our data as seen by the above graphs. For Dancability, we chose mean 0 and standard deviation 0.4 because the data has been standardized and should be centered around 0 and we chose 0.4 because it gave the best graph. We had tried multiple values for our standard deviation and ultimately settled on 0.4.

Note: since our data is standardized, we are assuming 0 is the correct mean for both priors.

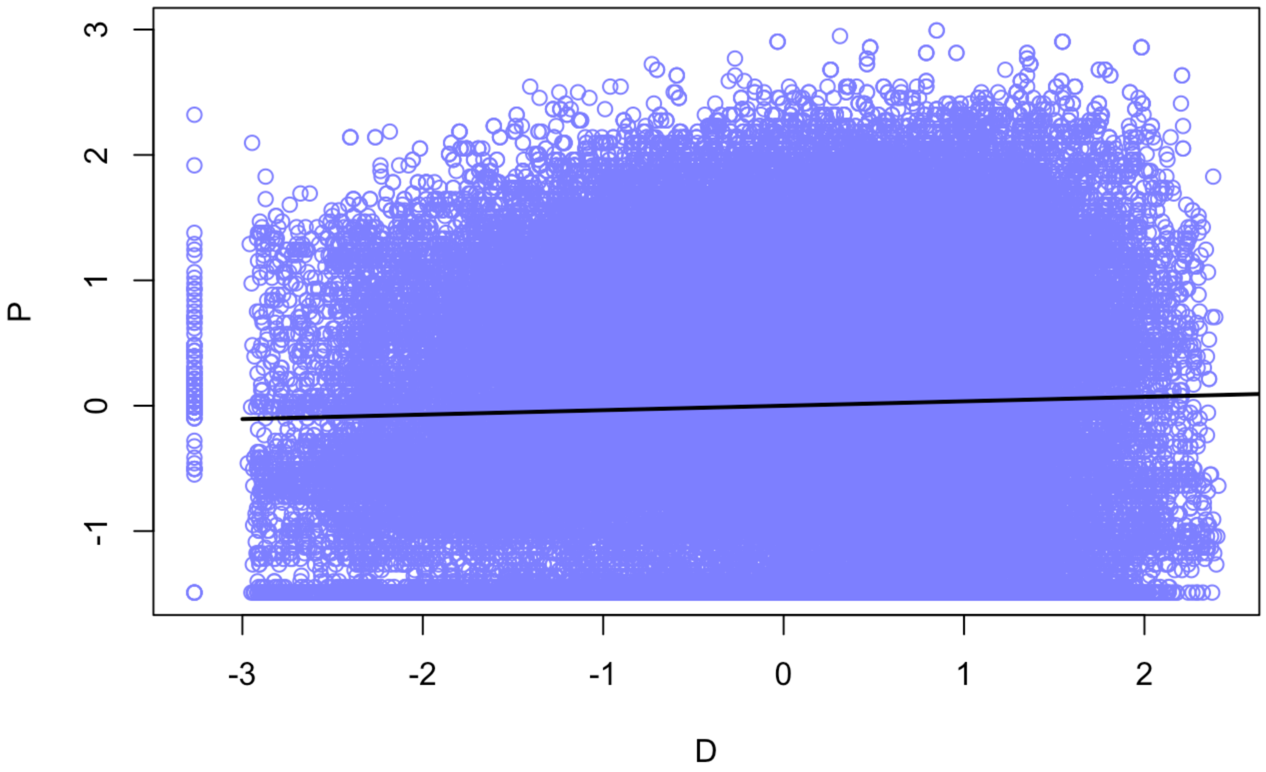
Simulating Posteriors

```
#Simulating the posterior for Explicit Model
E_seq <- seq(from = -3, to = 3.2, length.out = 30)
mu <- link(modelQ.1, data = list(E = E_seq))
mu.mean <- apply(mu, 2, mean)
mu.PI <- apply(mu, 2, PI)

#Plotting Explicit Model
plot(P ~ E, data = d, col = rangi2)
```

```
lines(E_seq, mu.mean, lwd = 2)
shade(mu.PI, E_seq)
```



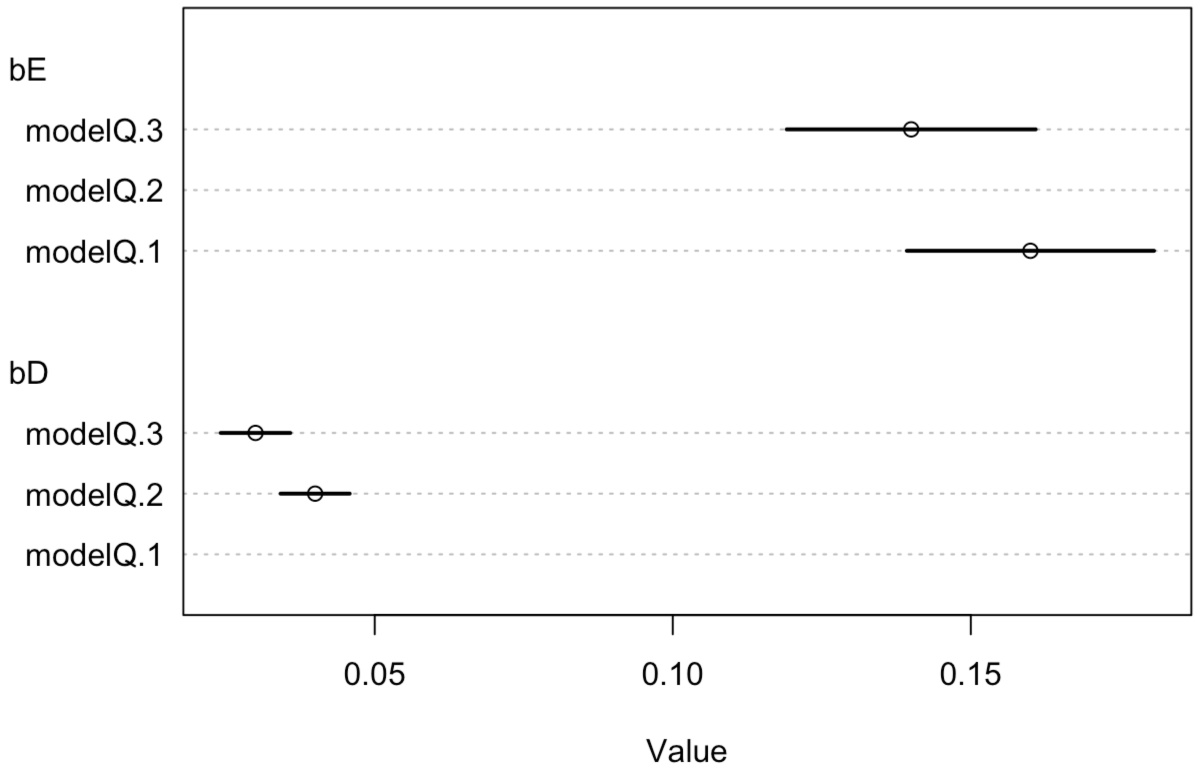
```
#Simulating the posterior for Dance Model
D_seq <- seq(from = -3, to = 3.2, length.out = 30)
mu <- link(modelQ.2, data = list(D = D_seq))
mu.mean <- apply(mu, 2, mean)
mu.PI <- apply(mu, 2, PI)

#Plotting Dance Model
plot(P ~ D, data = d, col = rangi2)
lines(D_seq, mu.mean, lwd = 2)
shade(mu.PI, D_seq)
```



Posteriors do not show promising results.

Plot Comparison Plot

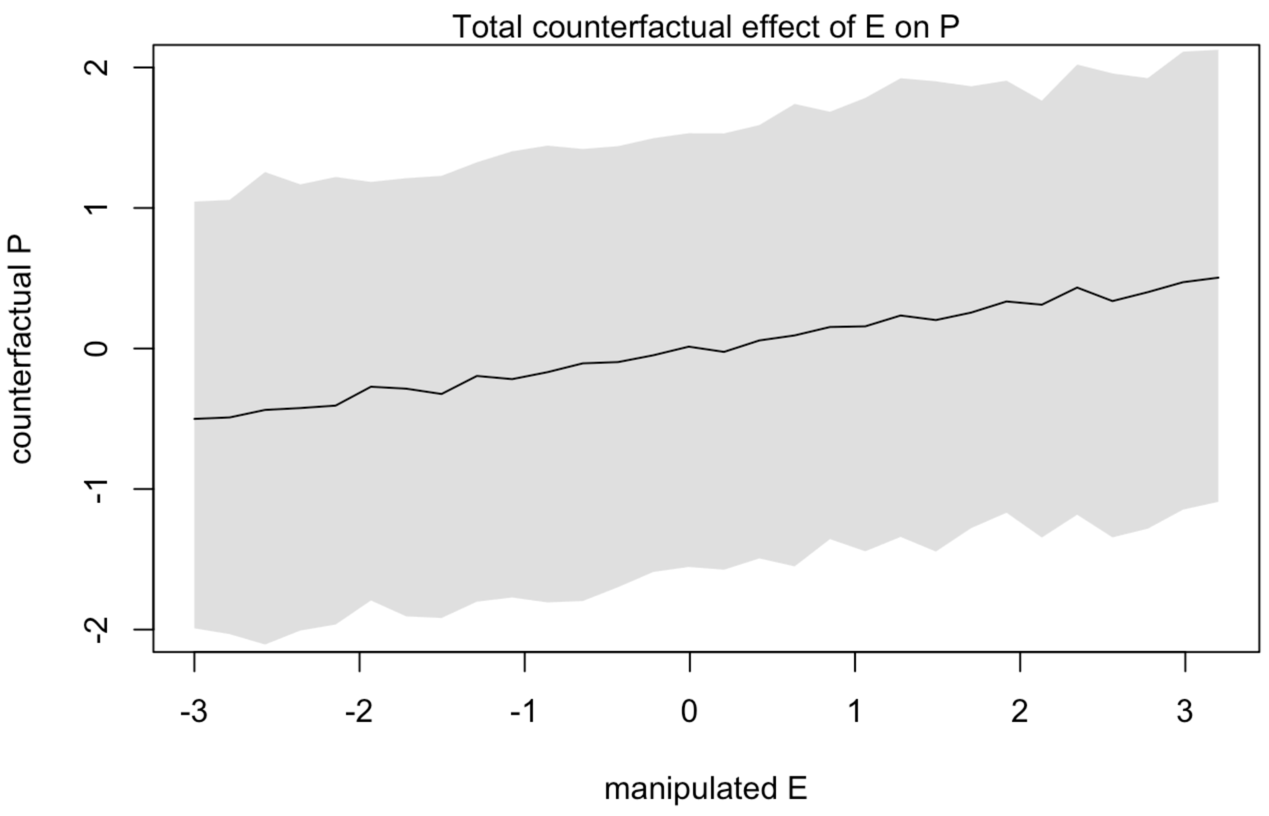
```
        plot(coeftab(modelQ.1, modelQ.2, modelQ.3), pars = c("bE", "bD"))
```



Building Counter-Factual Plot

```
modelQ.4 <- quap(
  alist(
    ## D -> P <- E
    P ~ dnorm(mu, sigma),
    mu <- a + bD * D + bE * E,
    a ~ dnorm(0, 0.2),
    bD ~ dnorm(0, 0.4),
    bE ~ dnorm(0, 0.5),
    sigma ~ dexp(1),

    ## E -> D
    D ~ dnorm(mu_D, sigma_D),
    mu_D <- aD + bED * E,
    aD ~ dnorm(0, 0.2),
    bED ~ dnorm(0, 0.5),
    sigma_D ~ dexp(1)
  ),
  data = d
)

D_seq <- seq(from = -2, to = 2, length.out = 30)
```

```
## R code 5.21
# prep data
sim_dat <- data.frame(E = E_seq)

# simulate M and then D, using A_seq
s <- sim(modelQ.4, data = sim_dat, vars = c("D", "P"))

## R code 5.22
plot(sim_dat$E, colMeans(s$P),
  ylim = c(-2, 2), type = "l",
  xlab = "manipulated E", ylab = "counterfactual P"
)
shade(apply(s$P, 2, PI), sim_dat$E)
mtext("Total counterfactual effect of E on P")
```



Looking at the above counter-factual plot showing the effect of a Song being Explicit on it's Danceability rating, we can conclude there is a very small relationship between the two variables. Although we are seeing a slight positive correlation, we also see in the shaded region a very low confidence in this relationship. We conclude there is not a masked relationship and will move forward without this relationship present.

Furthermore, our above Plot Comparison Plot shows that including both variables make a change in the weights of the two features of our model. However, both weights were pulled closer towards zero.

Neither include zero in their range, so we can interpret that both are useful in predicting popularity.

```
precis(modelQ.1)
```

```
             mean          sd        5.5%        94.5%
a     -0.01346945 0.003093633 -0.01841367 -0.008525226
bE     0.15757189 0.010578997  0.14066461  0.174479173
sigma  0.99901836 0.002092188  0.99567464  1.002362084
```

```
precis(modelQ.2)
```

```
               mean          sd         5.5%       94.5%
a     -1.310555e-06 0.002959535 -0.004731219 0.004728598
bD     3.544514e-02 0.002959791  0.030714818 0.040175454
sigma  9.993635e-01 0.002092916  0.996018623 1.002708390
```

```
precis(modelQ.3)
```

```
             mean          sd        5.5%        94.5%
a     -0.01232809 0.003094234 -0.01727328 -0.007382907
bD     0.03050767 0.002979853  0.02574528  0.035270046
bE     0.14421751 0.010654384  0.12718974  0.161245269
sigma  0.99856268 0.002091244  0.99522047  1.001904892
```

Looking into the precis results of our models, we can see there may be some over fitting occurring within our model. In all examples, the 89% HPDI is very small which means our model feels very certain in these numbers. This is a possible point of concern.

If we assume our model is not over fit, looking at these results we can interpret that a song with a higher danceablility score will be more popular and an explicit song with the same danceability as a non-explicit song, will be more popular.

Creating MCMC Models

Data Manipulation

```
d2 <- data

d2$P <- standardize(d$popularity)
d2$D <- standardize(d$danceability)
d2$E <- ifelse(d$explicit == "False", 0, 1)

sampled_d2 <- d2 %>% sample_n(10000)

dat_slim <- list(
  P = sampled_d2$P,
  D = sampled_d2$D,
  E = sampled_d2$E
)

str(dat_slim)
```

```
List of 3
 $ P: num [1:10000] 0.3031 -0.0107 0.5721 -0.5935 1.424 ...
  ..- attr(*, "scaled:center")= num 33.2
  ..- attr(*, "scaled:scale")= num 22.3
 $ D: num [1:10000] 0.433 1.73 -0.091 0.128 0.22 ...
  ..- attr(*, "scaled:center")= num 0.567
  ..- attr(*, "scaled:scale")= num 0.174
 $ E: num [1:10000] 1 1 0 0 0 0 0 0 0 0 ...
```

Our MCMC models take in data in the form of a list, so the data previously used for our Quadratic models needed to be manipulated for model consumption. Although the vessel is different, the content within is still the same.

```
# Predict Popularity with Explicit
modelMCM.1 <- ulam(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bE * E,
    a ~ dnorm(0, 0.2),
    bE ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ),
  data = dat_slim, cores = 4, chains = 4, log_lik = TRUE
)

# Predicting Popularity with Danceability
modelMCM.2 <- ulam(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bD * D,
    a ~ dnorm(0, 0.2),
    bD ~ dnorm(0, 0.40),
    sigma ~ dexp(1)
  ),
  data = dat_slim, cores = 4, chains = 4, log_lik = TRUE
)

# Predicting Popularity with Danceability and Explicit
modelMCM.3 <- ulam(
  alist(
    P ~ dnorm(mu, sigma),
    mu <- a + bD * D + bE * E,
    a ~ dnorm(0, 0.2),
    bD ~ dnorm(0, 0.4),
    bE ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ),
  data = dat_slim, cores = 4, chains = 4, log_lik = TRUE
)
```

Checking the models using Trankplots

Note: Due to needing to use the Log_Like parameter in order to compare the three models, traceplots will result in a plot for all 114,000 records for each model. For everyone's sake, this has been commented out, but the models were ran without the log_like functions and the chains all converged as expected.

```
#traceplot(modelMCM.1)

#traceplot(modelMCM.2)

#traceplot(modelMCM.3)
```

Creating Precis Tables

```
precis(modelMCM.1, depth=2)
```

```
             mean          sd        5.5%       94.5%     rhat ess_bulk
a     -0.01586091 0.010518852 -0.03249980 0.001287228 1.002750 1665.047
bE     0.12432729 0.038159111  0.06267363 0.187552130 1.000511 1664.730
sigma  1.00050514 0.007182468  0.98867176 1.012272200 1.002403 1697.908
```

```
precis(modelMCM.2, depth=2)
```

```
             mean          sd        5.5%      94.5%      rhat ess_bulk
a     -0.00622346 0.009926862 -0.02180381 0.00949016 0.9995821 1693.889
bD     0.02668950 0.009765527  0.01080115 0.04201366 1.0023833 1894.234
sigma  1.00048307 0.006808138  0.98937569 1.01148165 1.0004916 2005.917
```

```
precis(modelMCM.3, depth=2)
```

```
             mean          sd         5.5%       94.5%     rhat ess_bulk
a     -0.01504534 0.010744318 -0.032378850 0.002699323 1.000832 1920.328
bD     0.02346446 0.010019639  0.007181161 0.038712501 1.001579 2409.747
bE     0.11281966 0.036682486  0.054136475 0.171789490 1.003527 2120.960
sigma  1.00016552 0.007074081  0.988653590 1.011642200 1.000248 2719.886
```

Comparing the Precis results from our Quadratic Models and our MCMC Models, we see similar trends in the weights. However, now our 89% HDPI for danceability includes 0 meaning there is a chance that danceability does not influence popularity at all. Furthermore, the range in HPDI for Explicit has grown from .13-.16 to now .05-.16, showing a still confident and positive coefficient, but possibly not as large of one as seen before.

```
set.seed(42)
compare(modelMCM.1, modelMCM.2, modelMCM.3, func=WAIC)
```

```
               WAIC       SE    dWAIC      dSE    pWAIC     weight
modelMCM.3 28381.65 102.6625 0.000000       NA 3.749379 0.83549833
modelMCM.1 28385.25 102.6124 3.595902 4.642848 2.848418 0.13839022
modelMCM.2 28388.58 103.0785 6.931309 6.614771 2.432614 0.02611145
```

There is no real detectable difference between the models based on the WAIC values. ModelMCM.3 is the best as it has the lowest WAIC score, but they are all very similar to each other. Looking at the weight, there is a 85% certainty that modelMCM.3 is the best, so we will assume it is. However, since none of our models are very complex (2 feature models at most), they all have low pWAIC scores which can be a most of concern that we are missing possible predictors.

Of course, we knew this going into it. A song's popularity is not reasonably predicted using only danceability and whether or not it is explicit. There are several more factors that influence the popularity that are covered in this data set as well as many possible factors that are not include that may not be recorded anywhere.

Some further analysis with more features would be our next steps should we carry out this project.