HW5_GLMs

N. Barrus

2024-02-29

Purpose:

The purpose of this markdown document is to work through Homework 6 in Dr. Babcock's Bayesian Statistics Course at the University of Miami. Homework 6 deals with general lineral modelling.

General Start to Code

```
rm(list = ls())

######github#####
#note, only needed after 90 days from 1/16/2024

#  usethis::create_github_token()
#  gitcreds::gitcreds_set()

#####check for r updates#####
#note, updateing may take some time so plan accordingly

#require(installr)

#check.for.updates.R()

#updateR() #only if needed

#######check for package updates#####
#note, updateing may take some time so plan accordingly

#old.packages()

# update.packages() #make the decision to the update the packages
```

Load packages

```
library(tidyverse)
library(R2jags)
library(rstan)
library(ggmcmc)
```

```
library(purrr)
library(magrittr)
library(here)
library(loo)
library(DHARMa)
theme_set(theme_bw(base_size=15))
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Data

shark.csv is the presence or absence of accoustically tagged sharks at monitors in Belize, with habitat type (coded as integers) and the log distance (lnDist) from the shark's tagging location to the monitor.

mortality.csv is from a meta analysis of fish mortality rates by Amy Then et al., https://academic.oup.com/icesjms/article/72/1/82/2804320 the response variable is natural mortality (M), and the predictor variables are the growth rate (K), maximum length (Linf), longevity (tmax), and temperature (temp).

```
z.score <- function(.var) {
  z.score <- (.var-mean(.var))/sd(.var)
  z.score
}


shark.data <- read_csv(here("data","shark.csv")) |>
  mutate(z.lnDist = z.score(lnDist))
```

```
## Rows: 147 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): present, lnDist, habitat
##
## 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.
```

```
head(shark.data)



mortality.data <- read_csv(here("data","Mortality.csv")) |>
  mutate(z.K = z.score(K),
         z.Linf = z.score(Linf),
         z.tmax = z.score(tmax),
         z.temp = z.score(Temp))
```

```
## Rows: 215 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Genus, Species
## dbl (5): M, K, Linf, tmax, Temp
##
## 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.
```

```
head(mortality.data)
```

Problem 1: Binomial GLM/Logistic model of mortality

In this problem we will create a stan model to fit a logistic regression to the shark presence/absence data.

A) Fit a logistic regression model (similar to McCarthy box 5.8), with a logit link, to these data, with only the explanatory variable lnDist (minus its mean and divided by its standard deviation). Show the summary statistics for the regression coefficients and check for convergence. *write and run the model*

```
write("model
{
#uniformative priors
a ~ dnorm(0,1.0E-6)  #intercept term
b ~ dnorm(0,1.0E-6)  #regression coefficents


for (i in 1:N)
{
logit(p[i]) <- a + b*x[i]
y[i] ~ dbern(p[i])
resid[i] <- y[i]-p[i]

simval[i] ~dbern(p[i])

}
}
", file = here("JAGS_mods","HW6_A_logisticregression.txt"))

shark.data.list = list(y = shark.data$present,
                       x = shark.data$z.lnDist,
                       N = length(shark.data$present))

shark.jags <- jags(shark.data.list,parameters.to.save = c("p","a","b","resid","simval"),
                   model.file = here("JAGS_mods","HW6_A_logisticregression.txt"),
                   n.chains = 2, n.iter = 110000, n.burnin = 10000,
                   n.thin = 2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 147
##    Unobserved stochastic nodes: 149
##    Total graph size: 783
##
## Initializing model
##
##   |                                                      |
```

check coefficient summaries and convergence

```
range(shark.jags$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000988 1.002243
```

```
range(shark.jags$BUGSoutput$summary[,"n.eff"])
```

```
## [1]  13000 100000
```

```
shark.jags$BUGSoutput$summary[c("a","b"),]
```

```
##         mean        sd      2.5%       25%       50%       75%      97.5%     Rhat  n.eff
## a -0.5931259 0.2093885 -1.011066 -0.7319049 -0.5909585 -0.4508972 -0.1875577 1.001041  29000
## b -1.6749116 0.3046706 -2.308311 -1.8718078 -1.6632330 -1.4643069 -1.1125263 1.001001 100000
```

B) Now fit a model with lnDist and the habitat variable and their interaction, equivalent to glm(shark~lnDist*habitat) in R. Give the summaries of the parameters and check for convergence. *write and run the model*

```
write("model
{
#uniformative priors
a ~ dnorm(0,1.0E-6)  #intercept term

for(i in 1:3)
{
b[i] ~ dnorm(0,1.0E-6)  #regression coefficents
}

for (i in 1:N)
{
logit(p[i]) <- a + b[1]*x.lndist[i] + b[2]*x.hab[i] + b[3]*x.lndist[i]*x.hab[i]
y[i] ~ dbern(p[i])
resid[i] <- y[i]-p[i]

simval[i] ~dbern(p[i])

}
}
", file = here("JAGS_mods","HW6_B_logisticregression.txt"))

shark.data.list = list(y = shark.data$present,
                       x.lndist = shark.data$z.lnDist,
                       x.hab = shark.data$habitat,
                       N = length(shark.data$present))

shark.jags.2 <- jags(shark.data.list,parameters.to.save = c("p","a","b","resid","simval"),
                     model.file = here("JAGS_mods","HW6_B_logisticregression.txt"),
                     n.chains = 2, n.iter = 110000, n.burnin = 10000,
                     n.thin = 2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 147
##    Unobserved stochastic nodes: 151
##    Total graph size: 1023
##
## Initializing model
##
##   |                                                          |
```

check coefficient summaries and convergence

```
range(shark.jags.2$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000988 1.001117
```

```
range(shark.jags.2$BUGSoutput$summary[,"n.eff"])
```

```
## [1]  12000 100000
```

```
shark.jags.2$BUGSoutput$summary[c("a","b[1]","b[2]","b[3]"),]
```

```
##            mean        sd       2.5%        25%        50%        75%      97.5%     Rhat  n.eff
## a     0.5958587 0.5427853 -0.4376759  0.2250865  0.5852819  0.9527853  1.6875913 1.001030  36000
## b[1] -0.7422622 0.8261785 -2.3685468 -1.2921314 -0.7396604 -0.1896316  0.8828031 1.001004  98000
## b[2] -0.8733850 0.3768607 -1.6754434 -1.1105658 -0.8532390 -0.6107333 -0.1935851 1.001019  49000
## b[3] -0.7977728 0.5418189 -1.9449938 -1.1395940 -0.7669173 -0.4212043  0.1776874 1.000990 100000
```

C) Calculate WAIC for both models. Which model is preferable according to WAIC and what is deltaWAIC for the other model? *create WAIC table*

```
DIC.table <- tibble(model = c("LR-dist", "LR-dist*hab"),
                    model.ls = c(list(shark.jags), list(shark.jags.2))) |>
  mutate(DIC = map_dbl(model.ls, c(2,24)),
         pD = map_dbl(model.ls, c(2,23)),
         deltaDIC = DIC - min(DIC),
         weight = round(exp(-2*deltaDIC)/sum(exp(-2*deltaDIC)),digits = 5))

DIC.table |> select(-model.ls) |> knitr::kable(caption = "DIC Table")
```

Table 1: DIC Table

model	DIC	pD	deltaDIC	weight
LR-dist	147.0408	2.050601	2.429012	0.00771
LR-dist*hab	144.6118	4.274910	0.000000	0.99229

I was unsure how to calculate the LL for the model, so I was unable to select the best WAIC model, but the DIC model selected for the interaction model. The delta DIC was 2.397

```
n <- dim(shark.data)[1]
prows<-paste0("p[",1:n,"]")
shark.data$p <- shark.jags.2$BUGSoutput$summary[prows,"mean"]


confusion <- table(round(shark.data$p),shark.data$present)
confusion
```

D) For the WAIC best model, show the confusion matrix. What fraction of predictions were correct, assuming that a predicted probability of presence greater than 0.5 is predicted presence?

```
##
##      0  1
##   0 86 24
##   1  7 30
```

```
(confusion[1,1]+confusion[2,2])/sum(confusion)
```

```
## [1] 0.7891156
```

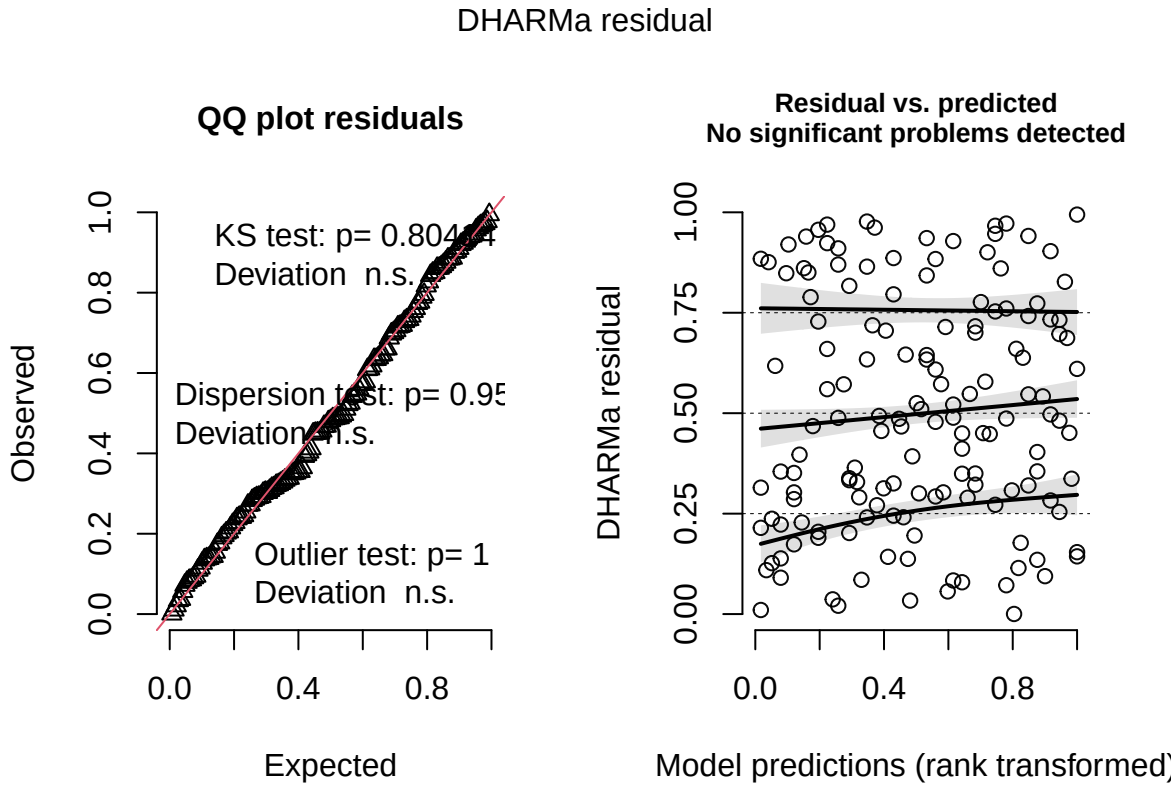
78.9% of predictions were correct.

```
#DHARMa residuals.
simval<-t(shark.jags.2$BUGSoutput$sims.matrix[,paste0("simval[",1:nrow(shark.data),"]")])
dim(simval)
```

E) For the WAIC best model, calculate the DHARMa residuals, and show the resulting plot. Does the model fit adequately?

```
## [1]    147 100000
```

```
simval<-simval[,sample(1:80000,500)]
DHARMaRes = createDHARMa(simulatedResponse =simval , observedResponse = shark.data$present,
             fittedPredictedResponse = shark.data$p, integerResponse = T)
plot(DHARMaRes)
```



The model fit looks adequate.

Problem 2: Normal and log normal

A) Using matrix format, set up a multiple regression to predict M from these 4 variables (just main effects, no interactions or quadratic terms). Use a normal likelihood. Give the regression coefficients, residual standard deviation, Bayesian P value, Rhat and n.eff. *set up matrix*

```
mortality.data

fit1 <- lm(M~z.K+z.Linf+z.tmax+z.temp, data = mortality.data)
mort.matrix <- model.matrix(fit1)

head(mort.matrix)
```

```
##   (Intercept)        z.K      z.Linf     z.tmax     z.temp
## 1           1  2.8959991 -0.78748126 0.25123314  1.0732645
## 2           1  0.6464902 -0.85501319 0.02927902  1.2367159
## 3           1 -0.6997632  0.54258472 1.69393493  0.2151445
## 4           1 -0.8666066  3.29752971 2.13784318 -1.1469507
## 5           1 -0.9212622  5.88029065 1.73092729 -0.8745316
## 6           1 -0.9298920  0.07093456 1.25002669 -1.4193697
```

write and run the model

```
write("model
      {
      #uniformative priors
      for(i in 1:Ncoef)
      {
      b[i] ~ dnorm(0,1.0E-6)
      }
      prec ~ dgamma(0.001, 0.001)

      #model
      for (i in 1:N)
      {
      ymean[i] <- inprod(b,xMatrix[i,])
      y[i] ~ dnorm(ymean[i],prec)

      pred.obs[i]~dnorm(ymean[i],prec)   # Predicted Y value
      resid[i]<-y[i]-ymean[i] #residuals
      sresid[i]<-(y[i]-ymean[i])*sqrt(prec) #standardized residual
      sresid2[i]<-sresid[i]*sresid[i]  #pearson residual squared
      rep.sresid2[i]<-(pred.obs[i]-ymean[i])*(pred.obs[i]-ymean[i])*prec
      LL[i]<--0.5*log(2*3.14159)+0.5*log(prec)-0.5*prec*(y[i]-ymean[i])*(y[i]-ymean[i])
      }

      #other quantities
      resid.sd <- sd(resid[])
      chi.square.obs<-sum(sresid2[])
      chi.square.rep<-sum(rep.sresid2[])
      p.value<-step(chi.square.obs-chi.square.rep)

      }", here("JAGS_mods","HW6_2a_matrixNormal.txt"))

mortality.list <- list(y = mortality.data$M,
                       xMatrix = mort.matrix,
                       Ncoef = 5,
                       N = length(mortality.data$M))

HW6_matrixNormal_jags <- jags(mortality.list,
                              model.file = here("JAGS_mods","HW6_2a_matrixNormal.txt"),
                              parameters.to.save = c("b","prec","resid.sd","p.value",
                                                     "resid","sresid","LL","chi.square.obs",
                                                     "chi.square.rep","ymean"),
                              n.chains=2,n.thin=10,n.iter=110000,n.burnin=10000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 215
##    Unobserved stochastic nodes: 221
##    Total graph size: 3471
##
## Initializing model
##
```

```
##   |                                                      |
```

```
range(HW6_matrixNormal_jags$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000948 1.001156
```

```
range(HW6_matrixNormal_jags$BUGSoutput$summary[,"n.eff"])
```

```
## [1]  7200 20000
```

```
HW6_matrixNormal_jags$BUGSoutput$summary[c("b[1]","b[2]","b[3]","b[4]","b[5]","resid.sd","p.value"),]
```

```
##                mean          sd       2.5%         25%         50%         75%       97.5%     Rhat
## b[1]      0.62545842 0.037949865  0.5506950  0.60009939  0.62562393  0.65067083  0.70057976 1.000983
## b[2]      0.46449798 0.045560675  0.3749855  0.43375610  0.46469798  0.49527663  0.55345916 1.001006
## b[3]     -0.05406759 0.042917583 -0.1384835 -0.08299739 -0.05403492 -0.02550590  0.03029295 1.000965
## b[4]     -0.15786692 0.043330500 -0.2424474 -0.18685860 -0.15790554 -0.12876078 -0.07260308 1.000963
## b[5]     -0.04496389 0.041470973 -0.1263823 -0.07289434 -0.04476589 -0.01702603  0.03594327 1.000952
## resid.sd  0.55738447 0.003742019  0.5527070  0.55463769  0.55655337  0.55926601  0.56667463 1.001098
## p.value   0.50175000 0.500009438  0.0000000  0.00000000  1.00000000  1.00000000  1.00000000 1.001096
##          n.eff
## b[1]     20000
## b[2]     20000
## b[3]     20000
## b[4]     20000
## b[5]     20000
## resid.sd 10000
## p.value  10000
```

```
write("model
      {
      #uniformative priors
      for(i in 1:Ncoef)
      {
      b[i] ~ dnorm(0,1.0E-6)
      }
      prec ~ dgamma(0.001, 0.001)

      #model
      for (i in 1:N)
      {
      ymean[i] <- inprod(b,xMatrix[i,])
      y[i] ~ dlnorm(ymean[i],prec)

      pred.obs[i]~dlnorm(ymean[i],prec)   # Predicted Y value
      resid[i]<-y[i]-ymean[i] #residuals
      sresid[i]<-(y[i]-ymean[i])*sqrt(prec) #standardized residual
      sresid2[i]<-sresid[i]*sresid[i]  #pearson residual squared
      rep.sresid2[i]<-(pred.obs[i]-ymean[i])*(pred.obs[i]-ymean[i])*prec
```

```
      LL[i]<--0.5*log(2*3.14159)+0.5*log(prec)-0.5*prec*(y[i]-ymean[i])*(y[i]-ymean[i])
      }

      #other quantities
      resid.sd <- sd(resid[])
      chi.square.obs<-sum(sresid2[])
      chi.square.rep<-sum(rep.sresid2[])
      p.value<-step(chi.square.obs-chi.square.rep)

      }", here("JAGS_mods","HW6_2b_matrixLogNormal.txt"))

mortality.list <- list(y = mortality.data$M,
                       xMatrix = mort.matrix,
                       Ncoef = 5,
                       N = length(mortality.data$M))

HW6_matrixLogNormal_jags <- jags(mortality.list,
                            model.file = here("JAGS_mods","HW6_2b_matrixLogNormal.txt"),
                            parameters.to.save = c("b","prec","resid.sd","p.value",
                                                   "resid","sresid","LL","chi.square.obs",
                                                   "chi.square.rep","ymean"),
                            n.chains=2,n.thin=10,n.iter=110000,n.burnin=10000)
```

B) Do the same model with a lognormal likelihood with everything else the same (i.e. predict the log scale mean from the linear model, logmean = x*b). Give the regression coefficients, residual standard deviation, Bayesian P value, Rhat and n.eff

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 215
##    Unobserved stochastic nodes: 221
##    Total graph size: 3471
##
## Initializing model
##
##   |                                                      |
##   |                                                      |
```

```
range(HW6_matrixLogNormal_jags$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000948 1.001372
```

```
range(HW6_matrixLogNormal_jags$BUGSoutput$summary[,"n.eff"])
```

```
## [1]  4800 20000
```

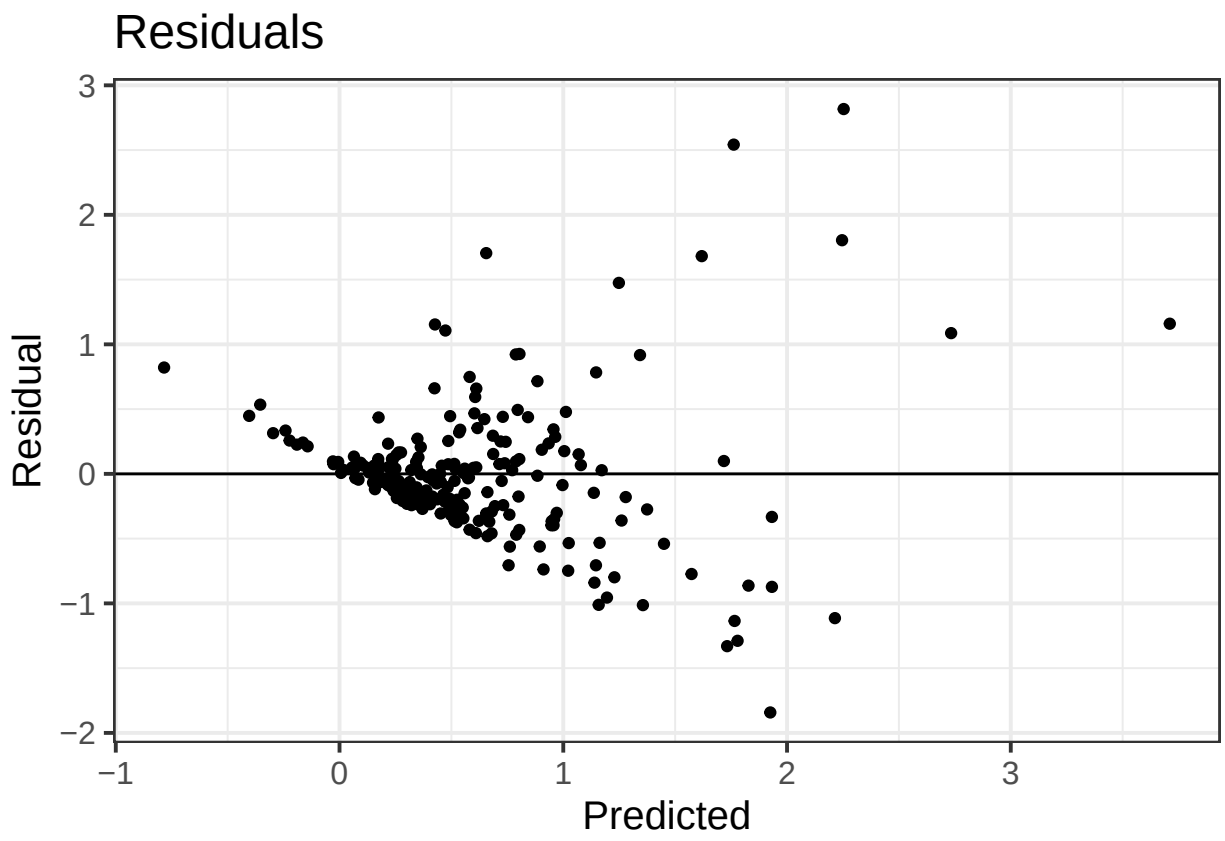
```
HW6_matrixLogNormal_jags$BUGSoutput$summary[c("b[1]","b[2]","b[3]","b[4]","b[5]","resid.sd","p.value"),]
```

```
##                mean         sd       2.5%        25%        50%        75%      97.5%     Rhat
```

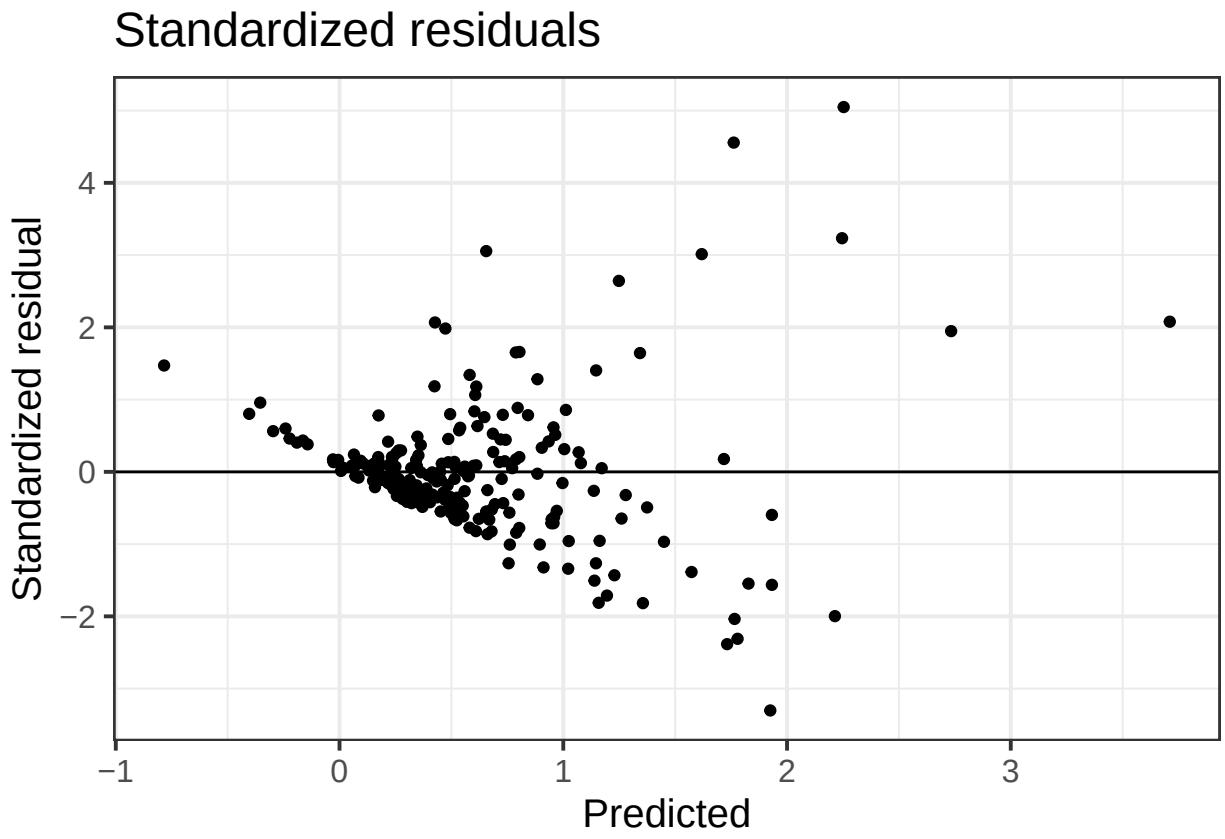
```
## b[1]     -1.02924700 0.04265149 -1.1121629 -1.0583444 -1.02939401 -1.00057700 -0.944872073 1.000976
## b[2]      0.28376231 0.05072856  0.1828769  0.2499212  0.28365011  0.31736579  0.384416531 1.000958
## b[3]     -0.09655267 0.04772456 -0.1905978 -0.1288033 -0.09624049 -0.06459496 -0.003734664 1.000955
## b[4]     -0.73561396 0.04813638 -0.8301223 -0.7678851 -0.73548558 -0.70329332 -0.641320916 1.000995
## b[5]     -0.07448829 0.04639004 -0.1653051 -0.1058935 -0.07442136 -0.04259829  0.015623852 1.001227
## resid.sd  0.77336463 0.02972509  0.7165539  0.7528641  0.77270720  0.79318128  0.832904720 1.000988
## p.value   0.63105000 0.48253242  0.0000000  0.0000000  1.00000000  1.00000000  1.000000000 1.000959
##          n.eff
## b[1]     20000
## b[2]     20000
## b[3]     20000
## b[4]     20000
## b[5]      5400
## resid.sd 20000
## p.value  20000
```

C) Plot the residuals for both models. Which seems to fit best? *plot the residuals for the normal model*

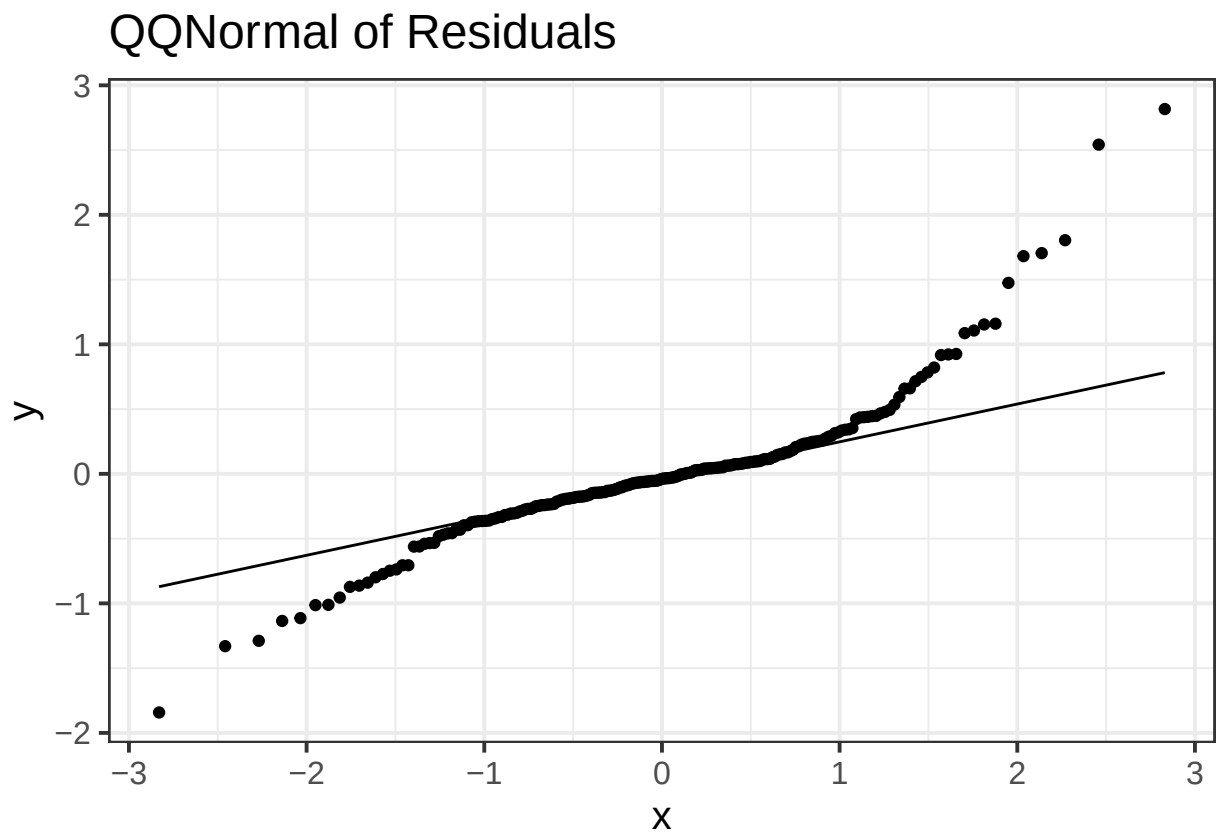
```
n<-length(mortality.data$M)
sumtab<-HW6_matrixNormal_jags$BUGSoutput$summary
residrows<-paste0("resid[",1:n,"]")
sresidrows<-paste0("sresid[",1:n,"]")
meanrows<-paste0("ymean[",1:n,"]")
dfcheck<-data.frame(Predicted=sumtab[meanrows,"mean"],
  Residual=sumtab[residrows,"mean"],
  SResid=sumtab[sresidrows,"mean"])
ggplot(dfcheck)+geom_point(aes(x=Predicted,y=Residual))+geom_abline(intercept=0,slope=0)+ggtitle("Residu
```



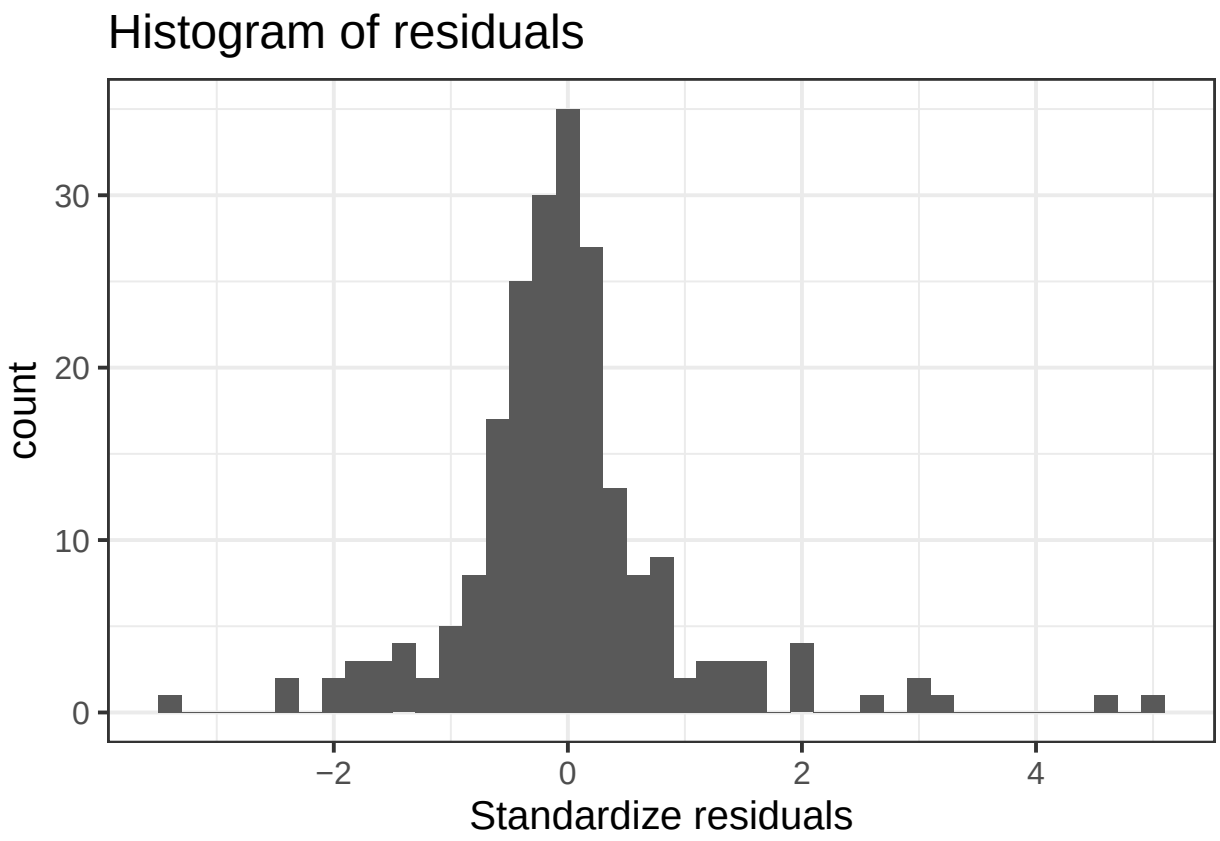

```
ggplot(dfcheck)+geom_point(aes(x=Predicted,y=SResid))+geom_abline(intercept=0,slope=0) +ylab("Standardi
```




```
ggplot(dfcheck,aes(sample=Residual))+geom_qq()+geom_qq_line()+ggtitle("QQNormal of Residuals")
```

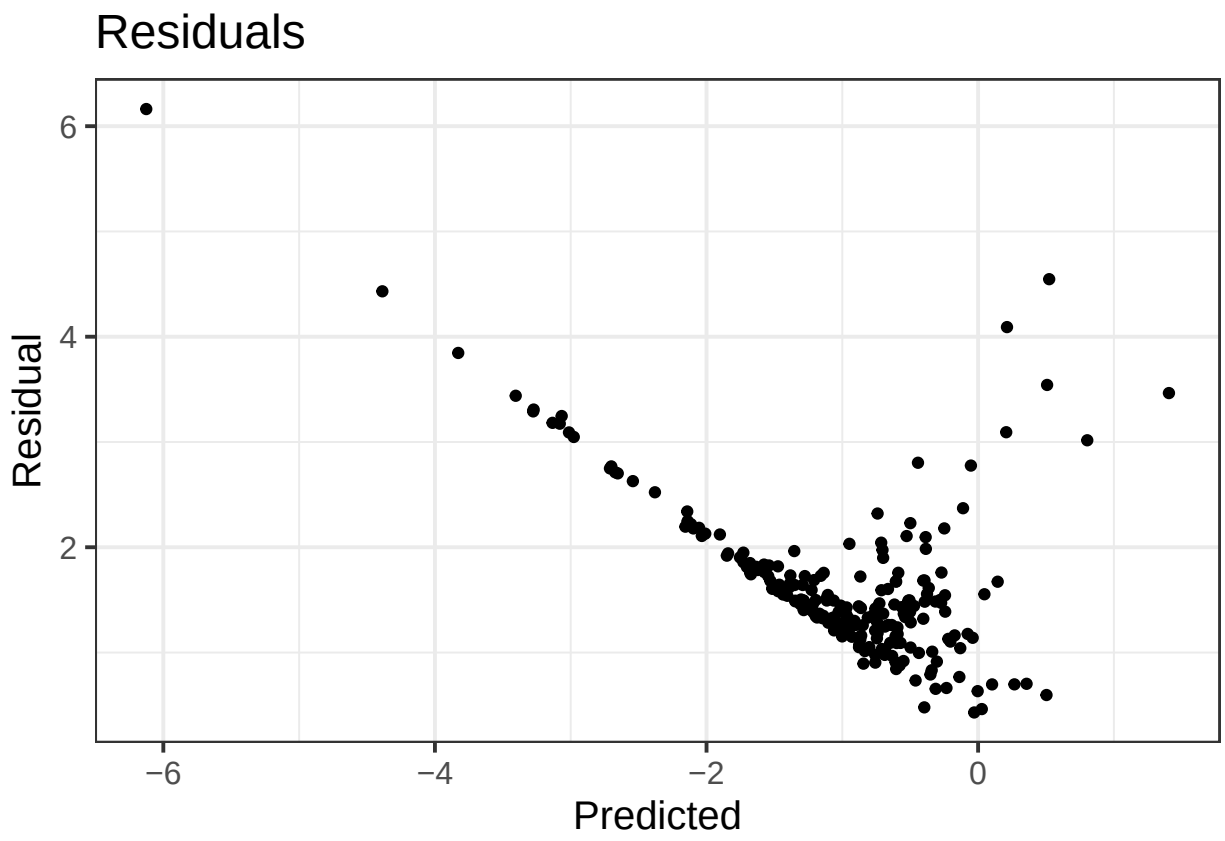



```
ggplot(dfcheck,aes(x=SResid))+geom_histogram(binwidth=.2)+xlab("Standardize residuals")+ggtitle("Histogr
```

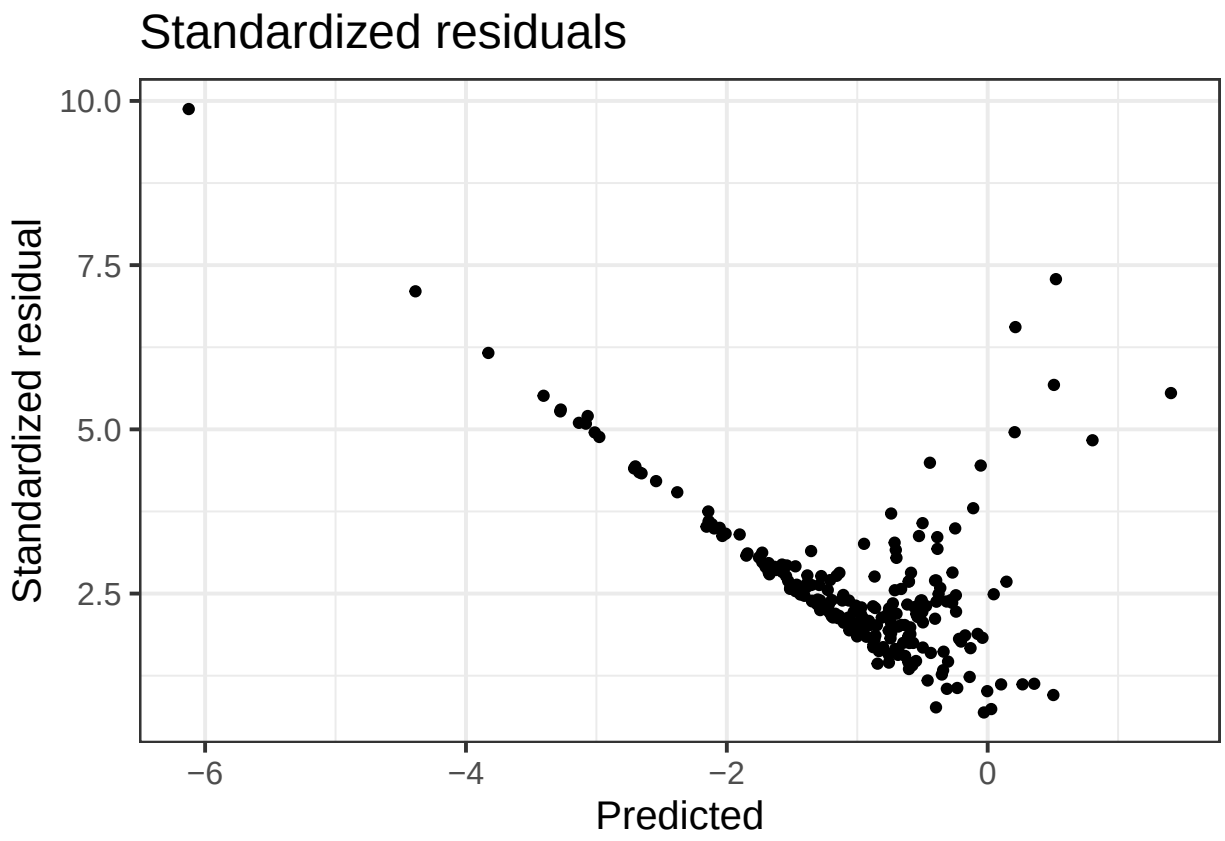


plot the residuald for the log normal model

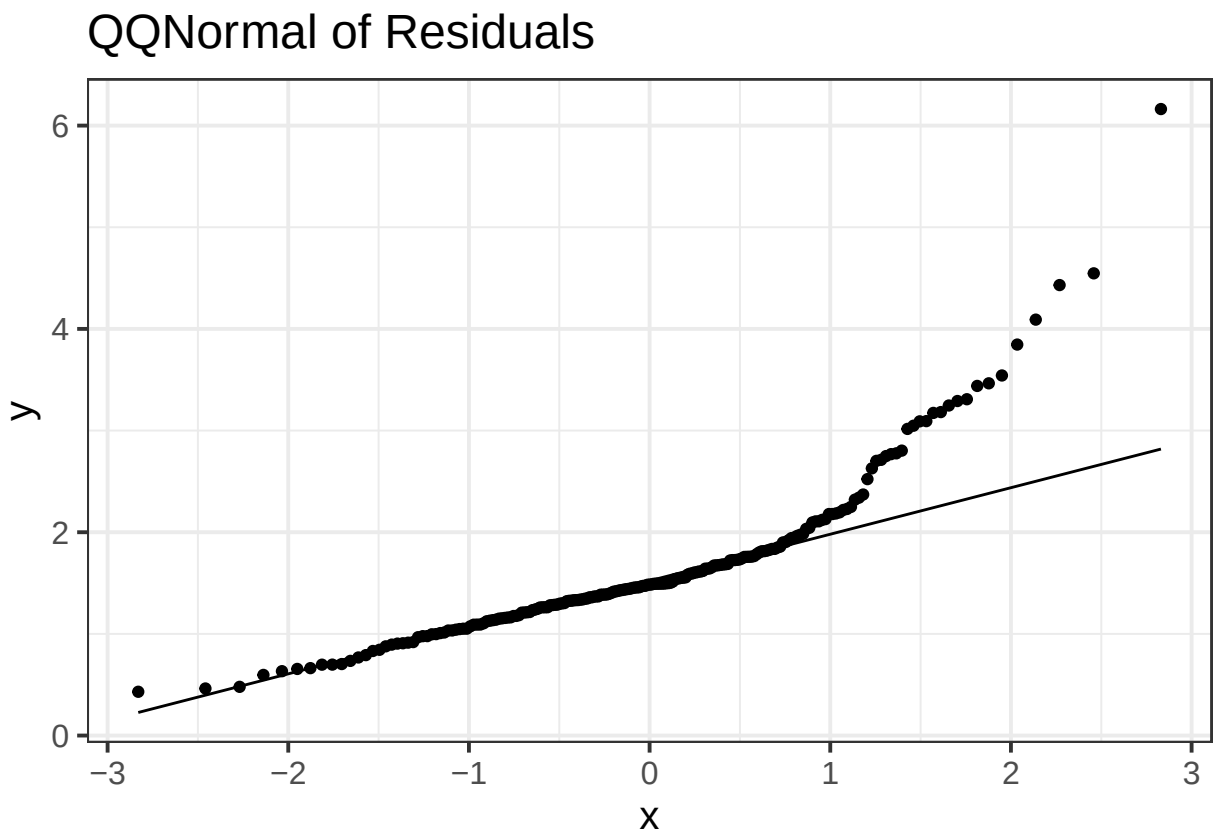
```
n<-length(mortality.data$M)
sumtab<-HW6_matrixLogNormal_jags$BUGSoutput$summary
residrows<-paste0("resid[",1:n,"]")
sresidrows<-paste0("sresid[",1:n,"]")
meanrows<-paste0("ymean[",1:n,"]")
dfcheck<-data.frame(Predicted=sumtab[meanrows,"mean"],
  Residual=sumtab[residrows,"mean"],
  SResid=sumtab[sresidrows,"mean"])
ggplot(dfcheck)+geom_point(aes(x=Predicted,y=Residual))+geom_abline(intercept=0,slope=0)+ggtitle("Residu
```



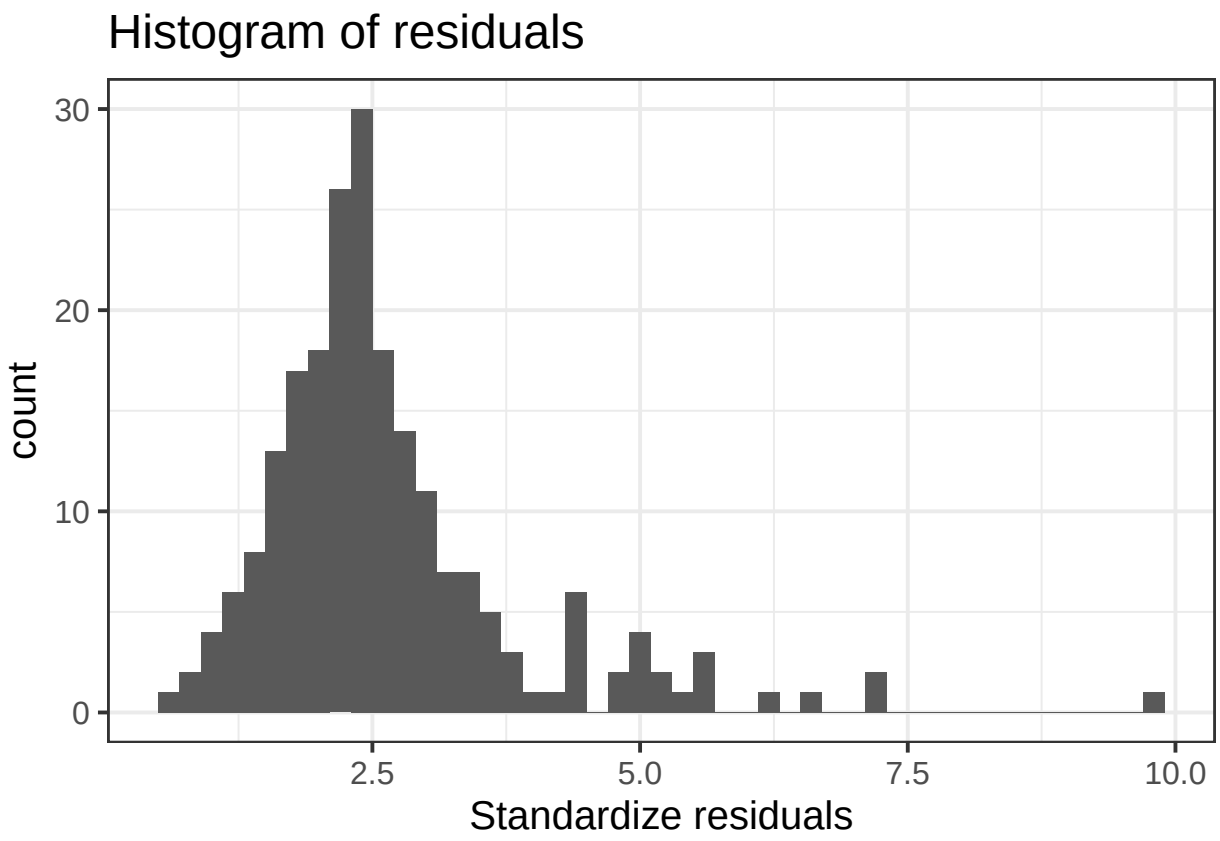

```
ggplot(dfcheck)+geom_point(aes(x=Predicted,y=SResid))+geom_abline(intercept=0,slope=0) +ylab("Standardi
```




```
ggplot(dfcheck,aes(sample=Residual))+geom_qq()+geom_qq_line()+ggtitle("QQNormal of Residuals")
```




```
ggplot(dfcheck,aes(x=SResid))+geom_histogram(binwidth=.2)+xlab("Standardize residuals")+ggtitle("Histogr
```



According to these residual plots that I have, the normal model was the best fitting model. But, I would note that I think that the way I set up the lognormal model was wrong because I would have expected the log nomormal model to improve the residuals based on the normal residual plot.

```
n<-length(mortality.data$M)
waic.table <- tibble(model = c("normal","lognormal"),
                     waic = c(            waic(HW6_matrixNormal_jags$BUGSoutput$sims.matrix[,paste0("LL
waic(HW6_matrixLogNormal_jags$BUGSoutput$sims.matrix[,paste0("LL[",1:n,"]")])$estimate[3,1])) |>
  mutate(deltWAIC = waic-min(waic),
         weight = round(exp(-2*deltWAIC)/sum(exp(-2*deltWAIC)),digits = 5))
```

D) Calculate WAIC for both models. Which model is preferable according to WAIC and what is deltaWAIC for the other model? Is the WAIC best model the same one that looked best from the residuals?

```
## Warning:
## 7 (3.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
## Warning:
## 44 (20.5%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic.table |> knitr::kable(caption = "WAIC Table")
```

Table 2: WAIC Table

model	waic	deltWAIC	weight
normal	372.8692	0.000	1
lognormal	2239.0704	1866.201	0

The normal model was preferred by using WAIC. It did reflect what I saw in the residual plots, but again I think the model specification was incorrect for the log normal model.

```
write("model
      {
      #uniformative priors
      for(i in 1:Ncoef)
      {
      b[i] ~ dnorm(0,1.0E-6)
      }
      prec ~ dgamma(0.001, 0.001)

      #model
      for (i in 1:N)
      {
      ymean[i] <- inprod(b,xMatrix[i,])
      logy[i] ~ dnorm(ymean[i],prec)

      #pred.obs[i]<-exp(ymean[i]+1/(2*prec))   # Predicted Y value
      #resid[i]<-y[i]-ymean[i] #residuals
      #sresid[i]<-(y[i]-ymean[i])*sqrt(prec) #standardized residual
     # sresid2[i]<-sresid[i]*sresid[i]  #pearson residual squared
     #rep.sresid2[i]<-(pred.obs[i]-ymean[i])*(pred.obs[i]-ymean[i])*prec
      #LL[i]<--0.5*log(2*3.14159)+0.5*log(prec)-0.5*prec*(y[i]-ymean[i])*(y[i]-ymean[i])
      }

      #other quantities
      #resid.sd <- sd(resid[])
      #chi.square.obs<-sum(sresid2[])
      #chi.square.rep<-sum(rep.sresid2[])
      #p.value<-step(chi.square.obs-chi.square.rep)

      }", here("JAGS_mods","HW6_2e_matrixLogY.txt"))

mortality.list <- list(y = mortality.data$M,
                       xMatrix = mort.matrix,
                       Ncoef = 5,
                       N = length(mortality.data$M))

#HW6_matrixLogNormal_jags <- jags(mortality.list,
#                                 model.file = here("JAGS_mods","HW6_2e_matrixLogY.txt"),
```

```
#                               parameters.to.save = c("b","prec","resid.sd","p.value",
#                                                      "resid","sresid","LL","chi.square.obs",
#                                                      "chi.square.rep","ymean"),
#                               n.chains=2,n.thin=10,n.iter=110000,n.burnin=10000)

#range(HW6_matrixLogNormal_jags$BUGSoutput$summary[,"Rhat"])
#range(HW6_matrixLogNormal_jags$BUGSoutput$summary[,"n.eff"])

#HW6_matrixLogNormal_jags$BUGSoutput$summary[c("b[1]","b[2]","b[3]","b[4]","b[5]","resid.sd","p.value")
```

E) Now run the normal model with log(M) as the response variable. Give the regression coefficients, residual standard deviation, Bayesian P value, Rhat and n.eff and WAIC. Do you get the same parameter values that you got with the lognormal likelihood in part b? Do you get the same WAIC? Why or why not? I am unsure how to specify this model as well, particularly when finding the predicted values. Becaue of this I was unable to provide the regression coefficients, residual standard deviation, Bayesian P value, Rhat and n.eff and WAIC. The coeficients should have been similar to the lognormal coeficients and the WAIC should have been smaller. But the smaller WAIC was incorrect because the likelihoods are different because the response variables is on different scales