HW9-GrowthModels&CatchIndices

Beth Babcock

2024-03-28

Purpose:

The purpose of this markdown document is to work through Homework 9 in Dr. Babcock's Bayesian Statistics Course at the University of Miami. Homework 9 deals with growth models and abundance indices.

General Start to Code

```r
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

```r
library(INLAutils)
library(INLA)
library(tidyverse)
library(R2jags)
library(rstan)
library(ggmcmc)
library(purrr)
library(magrittr)
library(here)
library(loo)
library(DHARMa)
library(lme4)
library(rstanarm)
library(shinystan)
library(BayesFactor)

theme_set(theme_bw(base_size=15))
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Data

For problem 1) The file croakerF.csv, shows the age and total length of 204 female croakers (a fish), from the croaker data set in the FSA R library.

For problem 2) The data in marlin.csv are counts of blue marlins observed in longline sets across 5 years. Remember to code the years as 1, 2, 3 and 4 for the input data to the JAGS or STAN model.

```r
#problem 1 data

croaker.data <- read_csv(here("data", "croakerF.csv"))
```

```
## Rows: 204 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): Age, ObsL
##
## 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.
```

```r
head(croaker.data)

#problem 2 data

marlin.data <- read_csv(here("data","marlin.csv")) |>
  mutate(present = if_else(Count > 0, true = 1, false = 0))
```

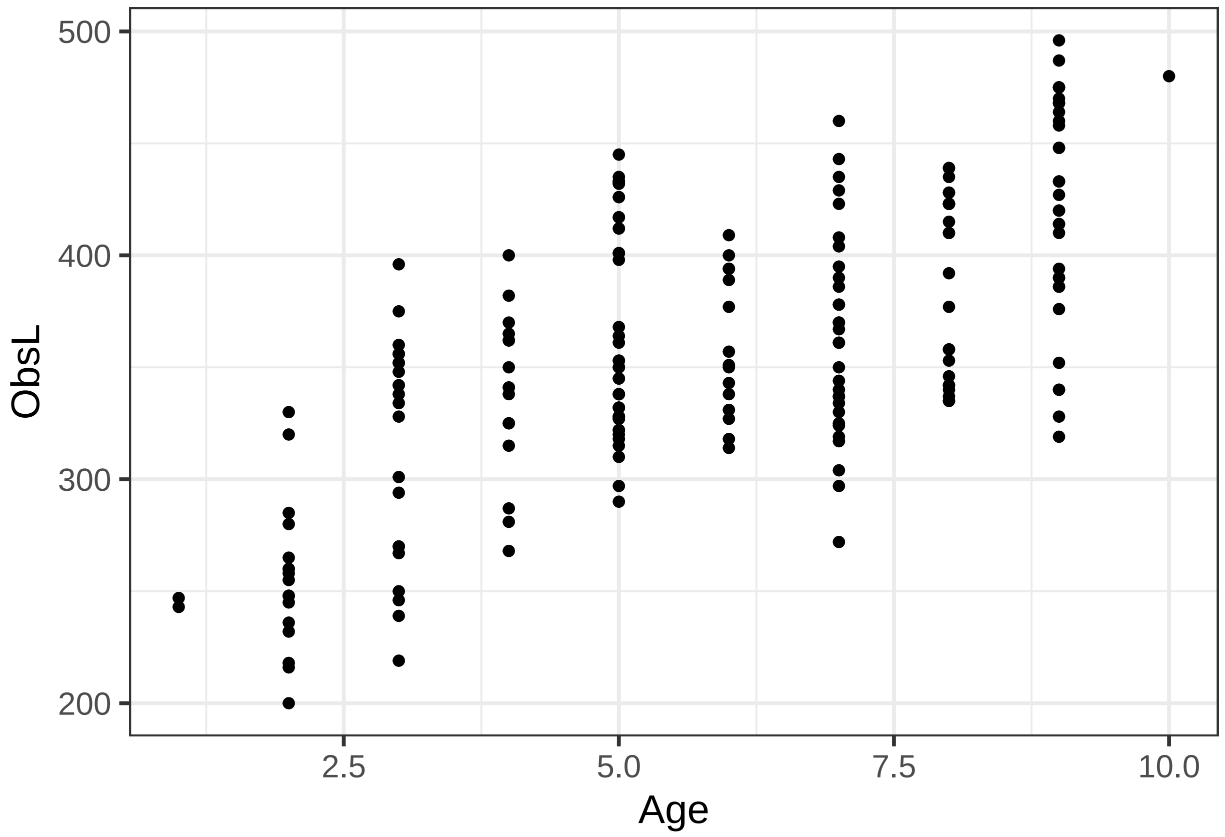
```
## Rows: 426 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): Count, Year
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(marlin.data)
```

Problem 1) Growth models

```r
croaker.data |>
  ggplot(aes(x = Age, y = ObsL))+
  geom_point()
```

A) Plot the length against age data. Is it obvious whether normal or lognormal will fit better?



It is not obvious whether the normal or lognormal will fit better. The error doesn't seem to increase with larger fish, so perhaps the normal is better.

```r
write("model
  {
        for (i in 1:N){
            # model prediction
            PredL[i] <- L1 +(L2-L1)* (1 - exp(- K* (Age[i] -Age1)))/(1 - exp(- K* (Age2 -Age1)))
            logPredL[i] <- log(PredL[i])     # log-transformation of pred.value
            ObsL[i] ~ dlnorm(logPredL[i], tau)       # lognormal likelihood
            logObsL[i] <-log(ObsL[i])        # log transfomration of observed value
             resid[i] <- logObsL[i]-logPredL[i]  # residuals
            Rep[i] ~ dlnorm(logPredL[i], tau) # replicated data set
            logRep[i] <-log(Rep[i]) # replicated data set
          sresid2[i]<-(logObsL[i]-logPredL[i])*(logObsL[i]-logPredL[i])*tau
      rep.sresid2[i]<-(logRep[i]-logPredL[i])*(logRep[i]-logPredL[i])*tau
      LL[i] <- -0.5*log(2*3.14159)+0.5*log(tau)-0.5*tau*(logObsL[i]-logPredL[i])^2-logObsL[i]
        }
  #priors specification
     K ~ dunif(0,2)
       Age1<-1
       Age2<-10
     L1~dunif(10,800)
     L2~dunif(10,800)
     tau~dgamma(0.001,0.001)
  #Derived parameters
    Linf<- (L2-L1*exp(-K*(Age2-Age1)))/(1-exp(-K*(Age2-Age1)))
    chi.square.obs<-sum(sresid2[])
    chi.square.rep<-sum(rep.sresid2[])
    p.value<-step(chi.square.obs-chi.square.rep)
    dev <- -2*sum(LL[])
}
",file=here("JAGS_mods","HW9_VonBertLnormSchnute_lognorm.txt"))

init1=list(tau=1,L1=200,L2=400,K=0.6)
init2=list(tau=0.5,L1=300,L2=450,K=0.4)

croaker.list <- list(N = length(croaker.data$Age),
                     Age = croaker.data$Age,
                     ObsL = croaker.data$ObsL)

HW9_SchnuteVBLG_lnorm_jags<-jags(croaker.list,list(init1,init2),
               model.file=here("JAGS_mods","HW9_VonBertLnormSchnute_lognorm.txt"),
              parameters.to.save=c("K","Linf","L1","L2","tau","p.value","LL","dev",
                "resid","PredL","logPredL", "chi.square.obs","chi.square.rep"),
              n.chains=2,n.iter=110000,n.burnin=10000,n.thin=10)
```

B) Fit a von Bertalanffy growth model using the Schnute formulation: with lognormal error and with uninformative priors. Set Age1 equal to 1, and Age2 equal to 10. Don't use any random effects. Note that this is a large data set, so don't save too many iterations (use a large thin) when you are saving residuals and predicted values.

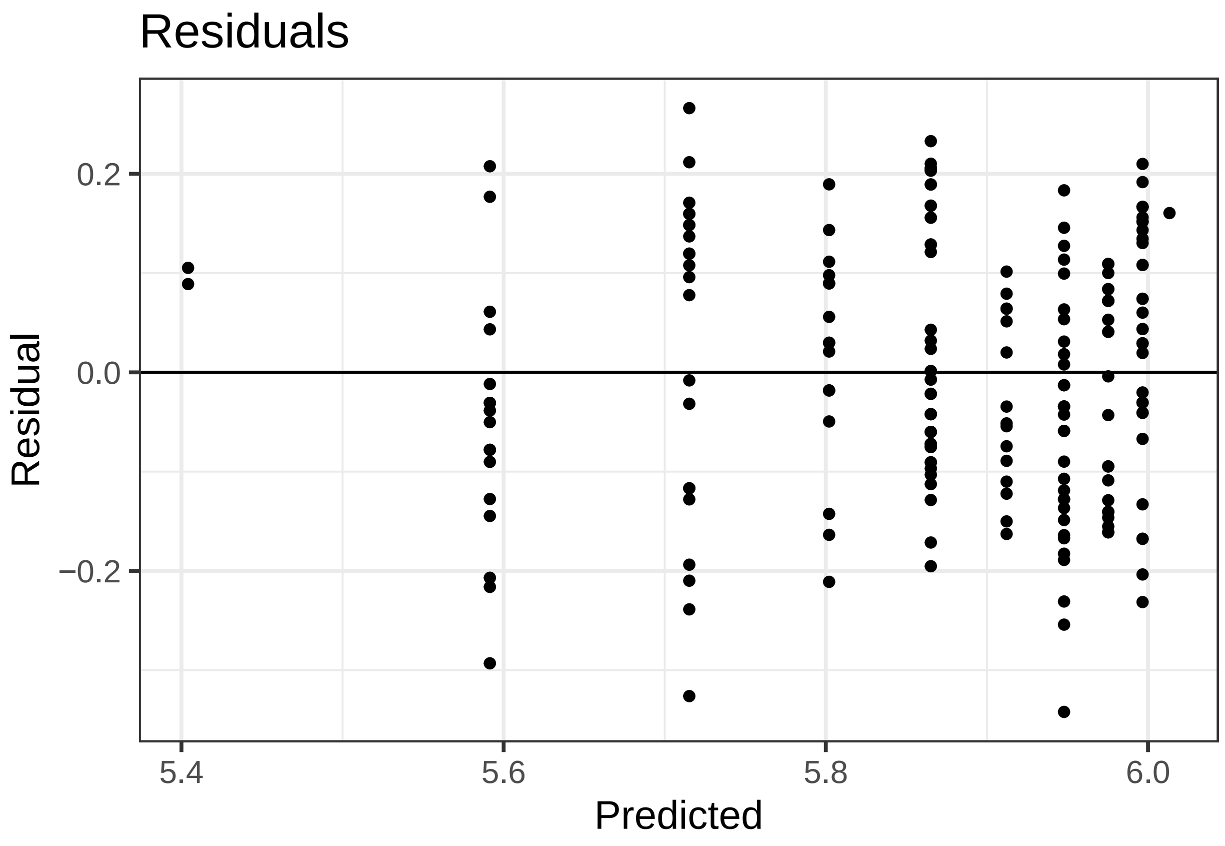
Show the summaries of the parameters (L1, L2, K and precision, and the derived quantity Linf), plots of residual against predicted value and the qqnormal plot of the residuals. Does the model appear to fit the data adequately? *summary*

```
parameters <- c("L1","L2","K","tau","Linf")

res2<-HW9_SchnuteVBLG_lnorm_jags$BUGSoutput

res2$summary[parameters,]
```

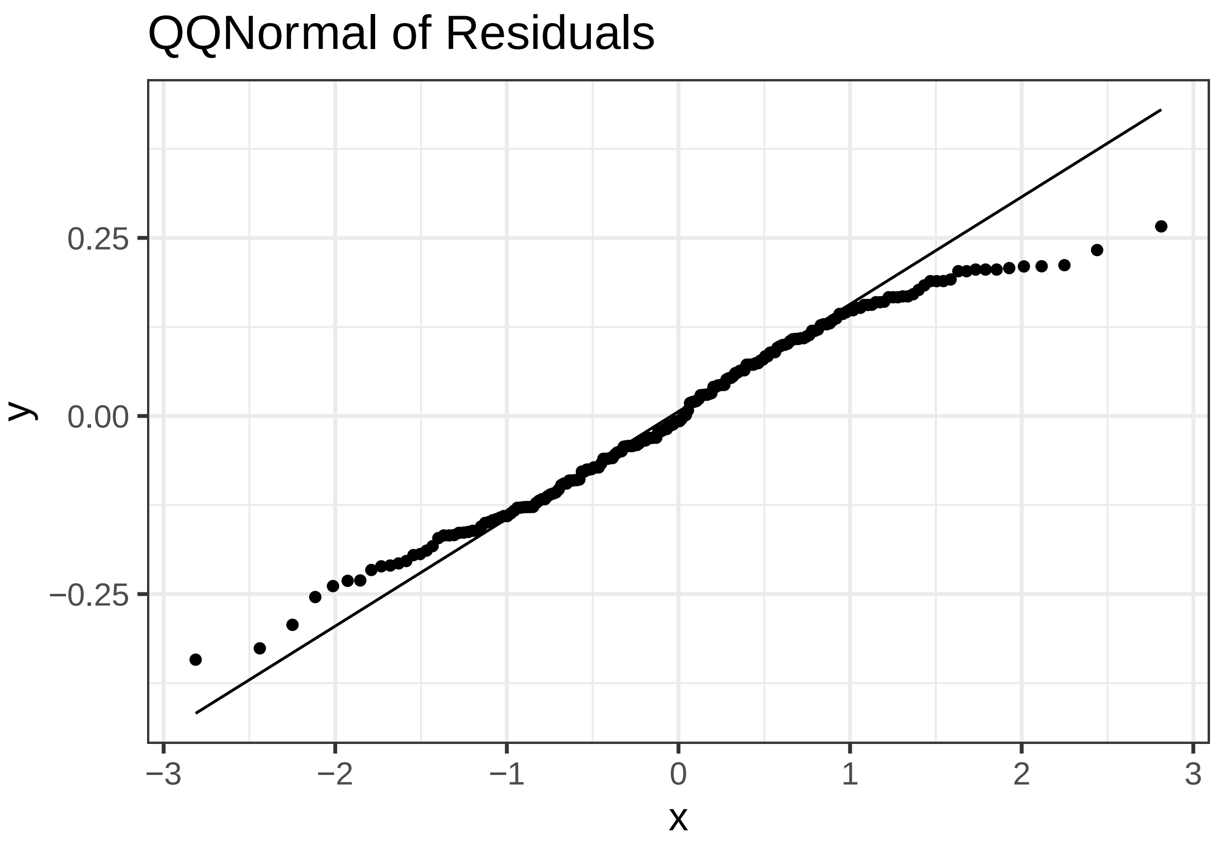
```
##                mean           sd         2.5%          25%          50%         75%
## L1   222.7343696  13.57278587 196.0457976 213.6355879 222.7248965 231.874055
## L2   408.9415508   9.03640053 392.5124838 402.6303717 408.5262048 414.707721
## K      0.2427689   0.06859581   0.1070472   0.1970037   0.2426753   0.288493
## tau   59.8985420   5.97267944  48.6780432  55.8058438  59.7223836  63.776630
## Linf 444.8457417 354.50636937 400.2130780 418.5608087 432.3842846 451.505572
##            97.5%     Rhat n.eff
## L1   249.3723566 1.001098 10000
## L2   427.9455704 1.000960 20000
## K      0.3790902 1.001108  9400
## tau   72.1606604 1.000953 20000
## Linf 536.9630098 1.001236 20000
```

residual plots

```
n<-length(croaker.data$Age)

sumtab<-res2$summary
residrows<-paste0("resid[",1:n,"]")
meanrows<-paste0("logPredL[",1:n,"]")
dfcheck<-data.frame(Predicted=sumtab[meanrows,"mean"],
  Residual=sumtab[residrows,"mean"])
ggplot(dfcheck)+
  geom_point(aes(x=Predicted,y=Residual))+
  geom_abline(intercept=0,slope=0)+ggtitle("Residuals")
```



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



C) Now fit the same model with normal error. Show the summaries of the parameters, plots of residual against predicted value and the qqnormal plot of the residuals. Does the model appear to fit the data adequately? *run the model*

```
write("model
  {
        for (i in 1:N){
            # model prediction
            PredL[i] <- L1 +(L2-L1)* (1 - exp(- K* (Age[i] -Age1)))/(1 - exp(- K* (Age2 -Age1)))
            ObsL[i] ~ dnorm(PredL[i], tau)      # lognormal likelihood
             resid[i] <- ObsL[i]-PredL[i]  # residuals
            Rep[i] ~ dnorm(PredL[i], tau) # replicated data set
          sresid2[i]<-(ObsL[i]-PredL[i])*(ObsL[i]-PredL[i])*tau
      rep.sresid2[i]<-(Rep[i]-PredL[i])*(Rep[i]-PredL[i])*tau
      LL[i] <- -0.5*log(2*3.14159)+0.5*log(tau)-0.5*tau*(ObsL[i]-PredL[i])*(ObsL[i]-PredL[i])
        }
  #priors specification
     K ~ dunif(0,2)
       Age1<-1
       Age2<-10
     L1~dunif(10,800)
     L2~dunif(10,800)
     tau~dgamma(0.001,0.001)
  #Derived parameters
    Linf<- (L2-L1*exp(-K*(Age2-Age1)))/(1-exp(-K*(Age2-Age1)))
    chi.square.obs<-sum(sresid2[])
    chi.square.rep<-sum(rep.sresid2[])
    p.value<-step(chi.square.obs-chi.square.rep)
    dev <- -2*sum(LL[])
}
",file=here("JAGS_mods","HW9_VonBertLnormSchnute_norm.txt"))

init1=list(tau=1,L1=200,L2=400,K=0.6)
init2=list(tau=0.5,L1=300,L2=450,K=0.4)

croaker.list <- list(N = length(croaker.data$Age),
                     Age = croaker.data$Age,
                     ObsL = croaker.data$ObsL)

HW9_SchnuteVBLG_norm_jags<-jags(croaker.list,list(init1,init2),
              model.file=here("JAGS_mods","HW9_VonBertLnormSchnute_norm.txt"),
             parameters.to.save=c("K","Linf","L1","L2","tau","p.value","LL",
               "resid","PredL", "chi.square.obs","chi.square.rep","dev"),
             n.chains=2,n.iter=110000,n.burnin=10000,n.thin=10)
```

model summary

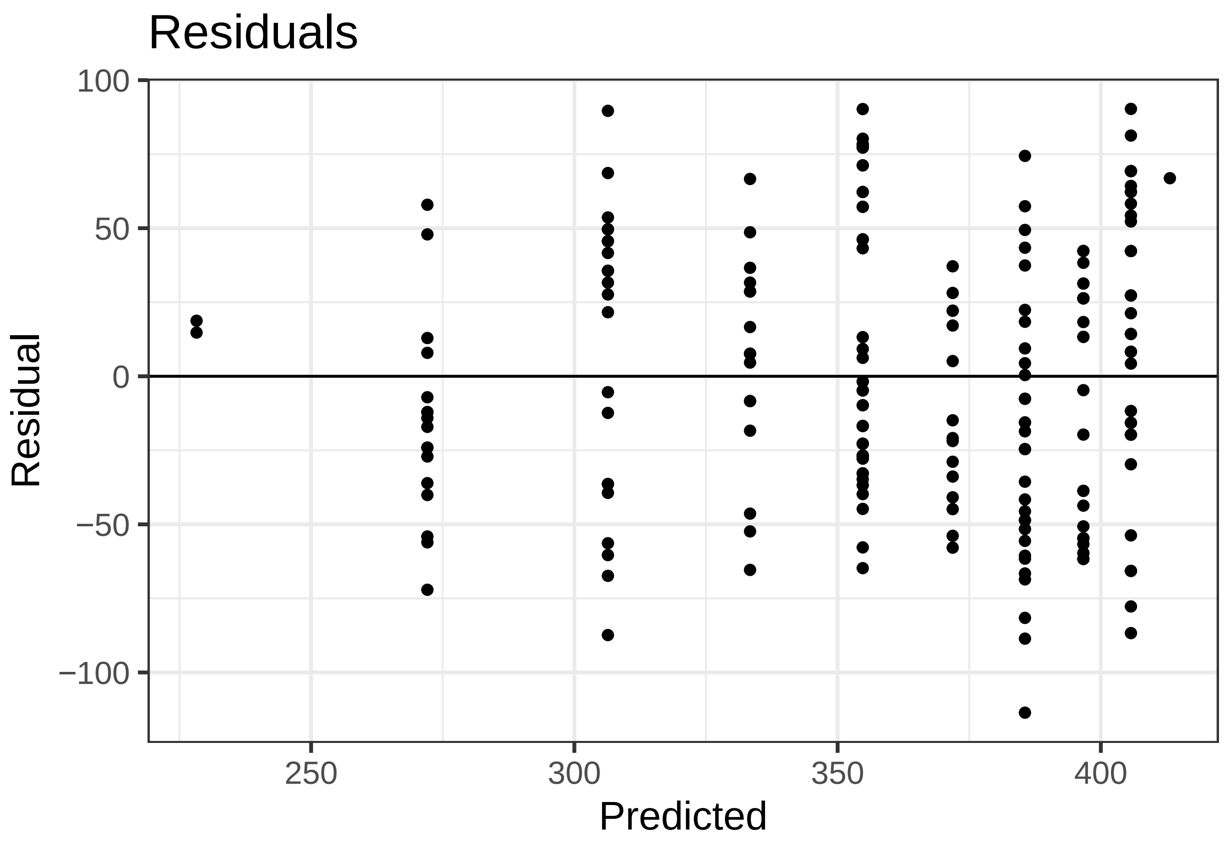
```
parameters <- c("L1","L2","K","tau","Linf")

res2_n<-HW9_SchnuteVBLG_norm_jags$BUGSoutput

res2_n$summary[parameters,]
```

```
##              mean           sd         2.5%          25%          50%
## L1   2.282425e+02 1.909225e+01 1.884407e+02 2.156532e+02 2.291927e+02
## L2   4.131163e+02 9.149843e+00 3.964258e+02 4.068167e+02 4.126032e+02
## K    2.279837e-01 8.302444e-02 6.888979e-02 1.704919e-01 2.267225e-01
## tau  4.810355e-04 4.742661e-05 3.921562e-04 4.483920e-04 4.793211e-04
## Linf 4.728097e+02 7.207494e+02 4.033281e+02 4.236028e+02 4.405706e+02
##               75%        97.5%     Rhat n.eff
## L1   2.417015e+02 2.630522e+02 1.001248  5000
## L2   4.191230e+02 4.320277e+02 1.001140  7800
## K    2.831192e-01 3.963893e-01 1.001125  8500
## tau  5.122972e-04 5.790449e-04 1.000986 20000
## Linf 4.673684e+02 6.282605e+02 1.001034 17000
```

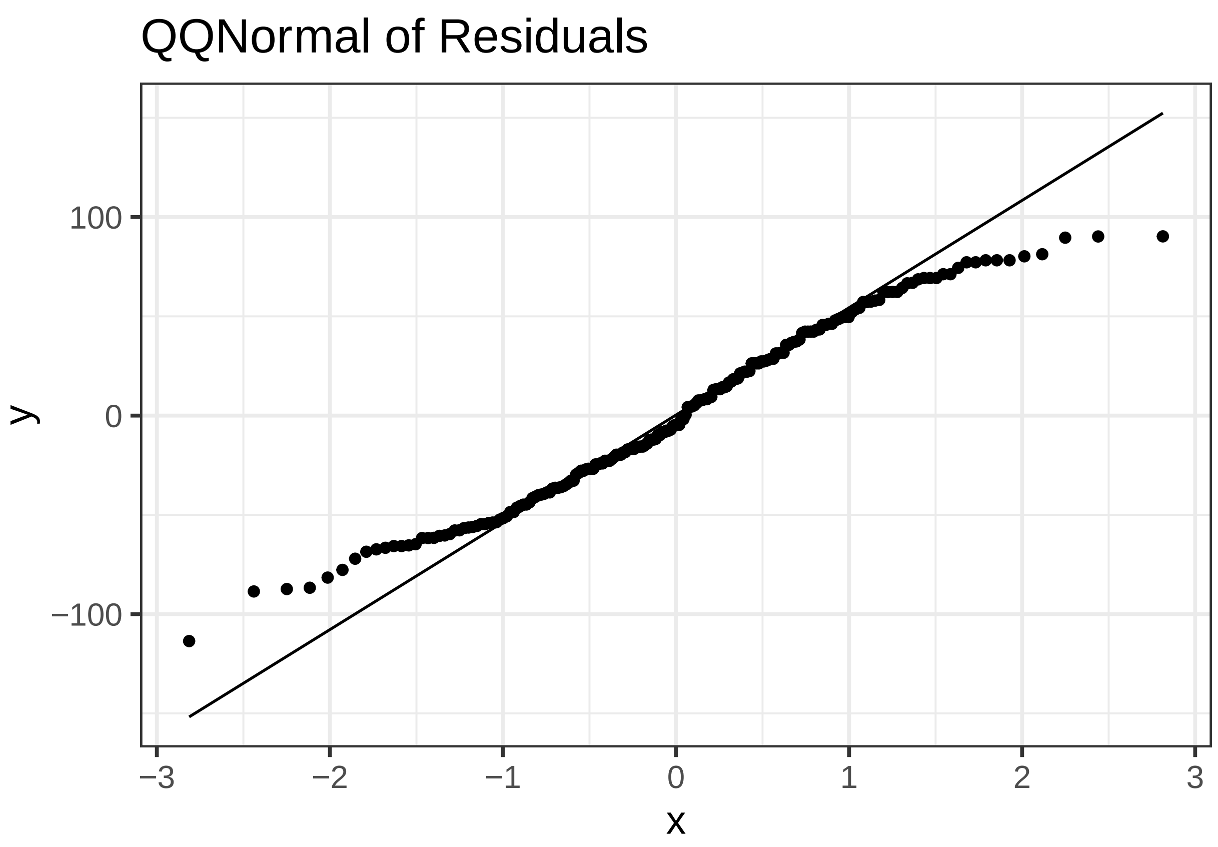
residual plots

```
n<-length(croaker.data$Age)

sumtab<-res2_n$summary
residrows<-paste0("resid[",1:n,"]")
meanrows<-paste0("PredL[",1:n,"]")
```

```r
dfcheck<-data.frame(Predicted=sumtab[meanrows,"mean"],
  Residual=sumtab[residrows,"mean"])
ggplot(dfcheck)+
  geom_point(aes(x=Predicted,y=Residual))+
  geom_abline(intercept=0,slope=0)+ggtitle("Residuals")
```



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



D) Compare the two models with WAIC. Is normal or lognormal better? (Hint: If you are using JAGS, you calculate the deviance as -2*sum(loglikelihood) to make sure you get the same deviance value that JAGS calculates. This is a way to check that you typed in the formula for loglikelihood correctly. STAN does this calculation of log likelihood for you so you don't need this. *check deviance*

```
#lognormal
HW9_SchnuteVBLG_lnorm_jags$BUGSoutput$summary[c("dev","deviance"),]
```

```
##              mean       sd     2.5%      25%      50%      75%    97.5%
## dev      2138.689 2.861634 2135.121 2136.582 2138.042 2140.097 2145.701
## deviance 2138.689 2.861634 2135.121 2136.582 2138.042 2140.097 2145.701
##              Rhat n.eff
## dev      1.000952 20000
## deviance 1.000952 20000
```

```
#normal
HW9_SchnuteVBLG_norm_jags$BUGSoutput$summary[c("dev","deviance"),]
```

```
##              mean       sd     2.5%      25%      50%      75%  97.5%     Rhat
## dev      2138.404 2.829896 2134.884 2136.318 2137.775 2139.818 2145.5 1.000954
## deviance 2138.404 2.829896 2134.884 2136.319 2137.775 2139.819 2145.5 1.000954
##          n.eff
## dev      20000
## deviance 20000
```

WAIC Table

```
n<-length(croaker.data$Age)

WAIC.table <- tibble(model = c("Lognormal", "normal"),
                     LL.ls = c(list(HW9_SchnuteVBLG_lnorm_jags$BUGSoutput$sims.matrix[,paste0("LL[",1:n
                     )|>
  mutate(WAIC.ls = map(LL.ls, waic),
         elpd_waic = map_dbl(WAIC.ls,c(1,1)),
         p_waic = map_dbl(WAIC.ls,c(1,2)),
         waic = map_dbl(WAIC.ls,c(1,3)),
         deltaWAIC = waic - min(waic),
         weight = round(exp(-2*deltaWAIC)/sum(exp(-2*deltaWAIC)),digits = 5))


WAIC.table |> select(-LL.ls,-WAIC.ls) |> knitr::kable(caption = "WAIC Table")
```

Table 1: WAIC Table

model	elpd_waic	p_waic	waic	deltaWAIC	weight
Lognormal	-1071.264	3.693358	2142.528	0.475448	0.2787
normal	-1071.026	3.566959	2142.052	0.000000	0.7213

Both the normal and lognormal are within 2 WAIC units so they both are adequate models. But, the normal model had the lowest WAIC.

E) Now add ageing error to the lognormal model, assuming that the input age is normally distributed around the true age with a standard deviation of 0.5. Give the true ages a uniform prior between 1 and 10. Show the summary statistics. How does this change the resulting estimates of Linf and K? (Note that you must use the estimated real age not the input age in the von Bertalanffy equation to predict length. Also, no need to calculate WAIC for this one). *run model*

```
write("model
  {
        for (i in 1:N){
          z[i] ~ dunif(1,10)
          Age[i] ~ dnorm(z[i],age_tau)
            # model prediction
            PredL[i] <- L1 +(L2-L1)* (1 - exp(- K* (z[i] -Age1)))/(1 - exp(- K* (Age2 -Age1)))
            logPredL[i] <- log(PredL[i])    # log-transformation of pred.value
            ObsL[i] ~ dlnorm(logPredL[i], tau)      # lognormal likelihood
            logObsL[i] <-log(ObsL[i])        # log transfomration of observed value
             resid[i] <- logObsL[i]-logPredL[i]  # residuals
            Rep[i] ~ dlnorm(logPredL[i], tau) # replicated data set
            logRep[i] <-log(Rep[i]) # replicated data set
          sresid2[i]<-(logObsL[i]-logPredL[i])*(logObsL[i]-logPredL[i])*tau
      rep.sresid2[i]<-(logRep[i]-logPredL[i])*(logRep[i]-logPredL[i])*tau
      LL[i] <- -0.5*log(2*3.14159)+0.5*log(tau)-0.5*tau*(logObsL[i]-logPredL[i])^2-logObsL[i]
        }
  #priors specification
```

```
     K ~ dunif(0,2)
       Age1<-1
       Age2<-10
     L1~dunif(10,800)
     L2~dunif(10,800)
     tau~dgamma(0.001,0.001)
  #Derived parameters
    age_tau <- 1/(0.5)^2
    Linf<- (L2-L1*exp(-K*(Age2-Age1)))/(1-exp(-K*(Age2-Age1)))
    chi.square.obs<-sum(sresid2[])
    chi.square.rep<-sum(rep.sresid2[])
    p.value<-step(chi.square.obs-chi.square.rep)
    dev <- -2*sum(LL[])
}
",file=here("JAGS_mods","HW9_VonBertLnormSchnute_lognorm_ageerror.txt"))

init1=list(tau=1,L1=200,L2=400,K=0.6)
init2=list(tau=0.5,L1=300,L2=450,K=0.4)

croaker.list <- list(N = length(croaker.data$Age),
                     Age = croaker.data$Age,
                     ObsL = croaker.data$ObsL)

HW9_SchnuteVBLG_lnorm.zage_jags<-jags(croaker.list,list(init1,init2),
              model.file=here("JAGS_mods","HW9_VonBertLnormSchnute_lognorm_ageerror.txt"),
             parameters.to.save=c("K","Linf","L1","L2","tau","p.value","LL","dev",
               "resid","PredL","logPredL", "chi.square.obs","chi.square.rep"),
             n.chains=2,n.iter=110000,n.burnin=10000,n.thin=10)
```

compare summaries

```
#lognormal
HW9_SchnuteVBLG_lnorm_jags$BUGSoutput$summary[c("Linf","K"),]
```

```
##             mean          sd        2.5%         25%         50%        75%
## Linf 444.8457417 354.50636937 400.2130780 418.5608087 432.3842846 451.505572
## K      0.2427689   0.06859581   0.1070472   0.1970037   0.2426753   0.288493
##            97.5%     Rhat n.eff
## Linf 536.9630098 1.001236 20000
## K      0.3790902 1.001108  9400
```

```
#lognormal with age error
HW9_SchnuteVBLG_lnorm.zage_jags$BUGSoutput$summary[c("Linf","K"),]
```

```
##             mean         sd        2.5%        25%         50%         75%
## Linf 426.9267524 30.20877098 394.3214792 410.184464 421.4373901 436.3379767
## K      0.2898854  0.07803393   0.1436877   0.237006   0.2867667   0.3398946
##            97.5%     Rhat n.eff
## Linf 491.5572409 1.001033 18000
## K      0.4516368 1.001113  9100
```

When including error in age, the Linf decreased and K increased when compared to the model that did not include error in age.

Problem 2) Delta lognormal

A) Apply a zero inflated Poisson model to the data. Use a logit link to estimate the probability of a positive observation as a function of the fixed effect of year and use a Bernoulli random variable (z) to estimate whether observations are positive or negative. Estimate the effect of year on the log of the mean count if positive as a fixed effect of year and use a Poisson likelihood for the counts. If you have trouble getting the model to run, try initializing the Bernoulli random variable for the positive catch (z in the example code) equal to 1 for all sets.

Show the summary statistics of the estimated parameters, making sure the model has converged. Plot the overall predicted mean count in each year from the model with credible intervals. *run the model*

```
write("model  {
  for(j in 1:Nyear) {
   a[j]~ dnorm(0, 1.0E-6)
   a2[j]~ dnorm(0, 1.0E-6)
 }
  for(i in 1:N)  {
   logit(p[i])<-a[Year[i]]
   z[i]~dbern(p[i])
   logMean[i]<-a2[Year[i]]
   Mu[i]<-z[i]*exp(logMean[i])
   count[i]~dpois(Mu[i])
  }
  for(j in 1:Nyear) {
   predmean[j] <- exp(a2[j])
   logit(predp[j])<-a[j]
   total.mean[j] <-predp[j] * predmean[j]
  }
}
",file=here("JAGS_mods","HW9_ZIP.txt"))

marlin.list <- list(N = length(marlin.data$present),
                    N2 = length(marlin.data$Count),
                    present = marlin.data$present,
                    Year = marlin.data$Year-1994,  #need the integers of year so the index in the loop
                    Year2 = marlin.data$Year-1994, #need the integers of year so the index in the loop
                    count = marlin.data$Count,
                    Nyear = length(unique(marlin.data$Year)))


init1 <- list(z = rep(1, times = nrow(marlin.data)))
init2 <- list(z = rep(1, times = nrow(marlin.data)))

HW9_ZIP_jags <- jags(marlin.list, inits = list(init1,init2),
                     parameters.to.save=c("a","a2","predmean","predp","total.mean"),
              model.file=here("JAGS_mods","HW9_ZIP.txt"),
              n.chains=2, n.iter=440000,
              n.burnin=40000,n.thin=8)
```

```
## Warning in jags.model(model.file, data = data, inits = init.values, n.chains =
## n.chains, : Unused variable "N2" in data
## Warning in jags.model(model.file, data = data, inits = init.values, n.chains =
## n.chains, : Unused variable "present" in data

## Warning in jags.model(model.file, data = data, inits = init.values, n.chains =
## n.chains, : Unused variable "Year2" in data
```

convergence, summary and plot

```
#convergence
range(HW9_ZIP_jags$BUGSoutput$summary[,"n.eff"])
```

```
## [1]     53 100000
```

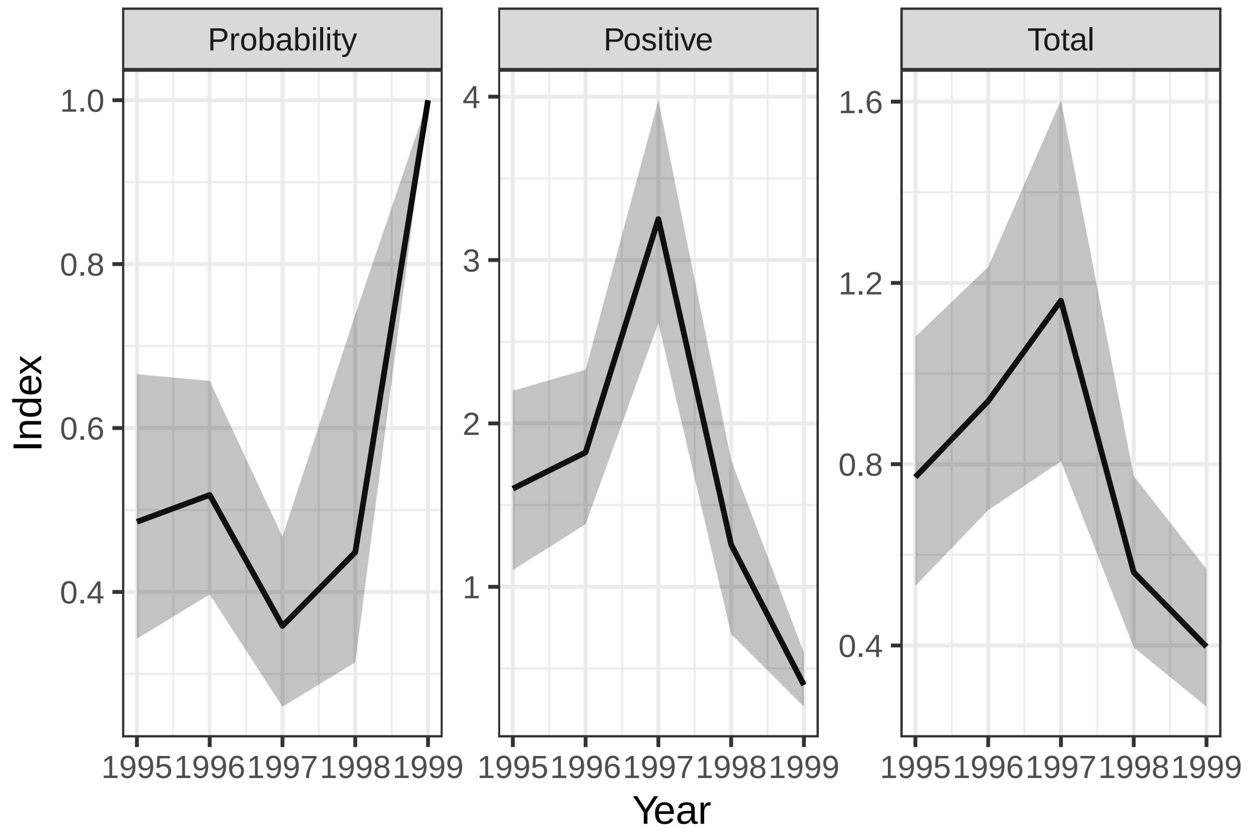
```
range(HW9_ZIP_jags$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000989 1.308417
```

```
#summary

HW9_ZIP_jags$BUGSoutput$summary
```

```
##                        mean           sd         2.5%          25%          50%
## a[1]            -0.03864770   0.34218218  -0.65045547  -0.26925856  -0.05794122
## a[2]             0.08342661   0.27166160  -0.41825253  -0.10192611   0.07330926
## a[3]            -0.58391278   0.23328377  -1.04640084  -0.74066485  -0.58187923
## a[4]             0.58784354   5.61481653  -0.78112066  -0.41590696  -0.20709814
## a[5]           784.65492947 606.01707596  12.34598266 302.86702765 658.96263970
## a2[1]            0.46327237   0.17576364   0.09723319   0.34971162   0.47038228
## a2[2]            0.59612667   0.13290749   0.32423642   0.50921372   0.60035097
## a2[3]            1.17679273   0.10694945   0.96102405   1.10615160   1.17895475
## a2[4]            0.20934672   0.22224769  -0.34081457   0.09440318   0.22989281
## a2[5]           -0.91800682   0.21392486  -1.32548028  -1.05365509  -0.91894109
## deviance       719.04022675  30.45652411 667.44802861 698.56588561 716.48425180
## predmean[1]      1.61366189   0.27998589   1.10211734   1.41865838   1.60060596
## predmean[2]      1.83104402   0.24143796   1.38297423   1.66398233   1.82275843
## predmean[3]      3.26247349   0.34752525   2.61437235   3.02270341   3.25097434
## predmean[4]      1.26158889   0.25789919   0.71119077   1.09900275   1.25846510
## predmean[5]      0.40903584   0.09829084   0.26567533   0.34866103   0.39894126
## predp[1]         0.49024853   0.08250885   0.34288691   0.43308913   0.48551875
## predp[2]         0.52036756   0.06637519   0.39693498   0.47454051   0.51831911
## predp[3]         0.35977435   0.05306291   0.25991684   0.32285878   0.35850030
## predp[4]         0.46472631   0.11366153   0.31407841   0.39749659   0.44840972
## predp[5]         0.99230248   0.06199873   0.99999565   1.00000000   1.00000000
## total.mean[1]    0.78069372   0.14027442   0.53147715   0.68246315   0.77152555
## total.mean[2]    0.94669816   0.13709234   0.69867877   0.85072750   0.93929282
## total.mean[3]    1.17190790   0.20365642   0.80642537   1.02820771   1.16118873
## total.mean[4]    0.56733003   0.09636125   0.39583529   0.49949734   0.56148168
## total.mean[5]    0.40241561   0.07772672   0.26479109   0.34762662   0.39726921
##                         75%        97.5%     Rhat  n.eff
## a[1]             0.16813085    0.6883908 1.000990 100000
## a[2]             0.25574457    0.6522434 1.001020  47000
## a[3]            -0.42506392   -0.1308883 1.000989 100000
## a[4]             0.02382918    1.0378461 1.308417     53
## a[5]          1141.86895010 2233.2926297 1.001174   8100
## a2[1]            0.58471938    0.7884962 1.001016  56000
## a2[2]            0.68752073    0.8452137 1.001006  86000
## a2[3]            1.25036090    1.3809188 1.001000 100000
## a2[4]            0.35515090    0.5738839 1.044353    150
## a2[5]           -0.78796027   -0.5184202 1.002646   3300
## deviance       736.06740390  790.5806883 1.021404    180
## predmean[1]      1.79448734    2.2000853 1.001016  56000
## predmean[2]      1.98877870    2.3284753 1.001006  86000
## predmean[3]      3.49160285    3.9785553 1.001000 100000
## predmean[4]      1.42639588    1.7751481 1.044353    150
## predmean[5]      0.45477146    0.5954605 1.002646   3300
## predp[1]         0.54193398    0.6656089 1.000990 100000
## predp[2]         0.56358993    0.6575158 1.001026  40000
## predp[3]         0.39530563    0.4673246 1.000989 100000
## predp[4]         0.50595701    0.7384342 1.050746    130
## predp[5]         1.00000000    1.0000000 1.072033    470
## total.mean[1]    0.86935427    1.0814194 1.001003 100000
## total.mean[2]    1.03501006    1.2352646 1.000991 100000
## total.mean[3]    1.30252202    1.6032369 1.000991 100000
## total.mean[4]    0.62851073    0.7739481 1.000989 100000
## total.mean[5]    0.45158067    0.5682931 1.001001 100000
```

```r
#plot
model.bugsoutput <- HW9_ZIP_jags$BUGSoutput
Nyear= marlin.list$Nyear
rows.p1<-paste0("predp[",1:Nyear,"]")
rows.ln1<-paste0("predmean[",1:Nyear,"]")
rows.tot1<-paste0("total.mean[",1:Nyear,"]")
df1<-data.frame(model.bugsoutput$summary[rows.p1,c("2.5%","50%","97.5%")])
df3<-data.frame(model.bugsoutput$summary[rows.ln1,c("2.5%","50%","97.5%")])
df5<-data.frame(model.bugsoutput$summary[rows.tot1,c("2.5%","50%","97.5%")])
resdf1<-bind_rows(list(Probability=df1,Positive=df3,Total=df5),.id = "Type")%>%
  mutate(Type=factor(Type,levels=c("Probability","Positive","Total")))
names(resdf1)[2:4]<-c(c("min","med","max"))
resdf1$Year<-rep(sort(unique(marlin.data$Year)),3)

#Plot
ggplot(resdf1,aes(x=Year,y=med,ymin=min,ymax=max))+
  geom_line(linewidth=1)+
  geom_ribbon(alpha=0.3)+
  facet_wrap(Type~.,scale="free")+
  ylab("Index")
```



B) Fit an integrated (i.e. all in one chunk of JAGS or STAN code) delta-lognormal model to these data, with the same model structure, i.e a fixed effect of all five years in both the binomial and lognormal parts of the model. Show the summary statistics of the estimated parameters, making sure the model has converged. Plot the overall predicted mean count in each year from the model with credible intervals. *run the model*

```
write("model  {
  tau~dgamma(0.1,0.001)
  for(i in 1:Nyear) {
   a[i]~ dnorm(0, 1.0E-6)
   a2[i]~ dnorm(0, 1.0E-6)
 }
  for(i in 1:N)  {
   logit(p[i])<-a[Year[i]]
   present[i]~dbern(p[i])
  }
  for(i in 1:N2)  {
   logMean[i]<-a2[Year2[i]]
   count[i]~dlnorm(logMean[i],tau)
  }
  for(i in 1:Nyear) {
   mean.lnorm[i] <- exp(a2[i]+1/(2*tau))
   logit(p.year[i])<-a[i]
   total.mean[i] <-p.year[i]* mean.lnorm[i]
  }
}
",file=here("JAGS_mods","HW9_DeltaLognormalGLM.txt"))


marlin.list <- list(N = length(marlin.data$present),
                    N2 = length(marlin.data$Count[marlin.data$present == 1]),
                    present = marlin.data$present,
                    Year = marlin.data$Year-1994,  #need the integers of year so the index in the loop
                    Year2 = marlin.data$Year[marlin.data$present == 1]-1994, #need the integers of year
                    count = marlin.data$Count[marlin.data$present == 1],
                    Nyear = length(unique(marlin.data$Year)))

HW9_deltaLnorm_jags <-jags(marlin.list,
  parameters.to.save=c("a","a2","mean.lnorm","p.year",
                       "total.mean"),
  model.file=here("JAGS_mods","HW9_DeltaLognormalGLM.txt"),
  n.chains=2, n.iter=100000, n.burnin=5000,n.thin=2)
```

convergence, summary and plot

```
#convergence
range(HW9_deltaLnorm_jags$BUGSoutput$summary[,"n.eff"])
```

```
## [1] 12000 95000
```

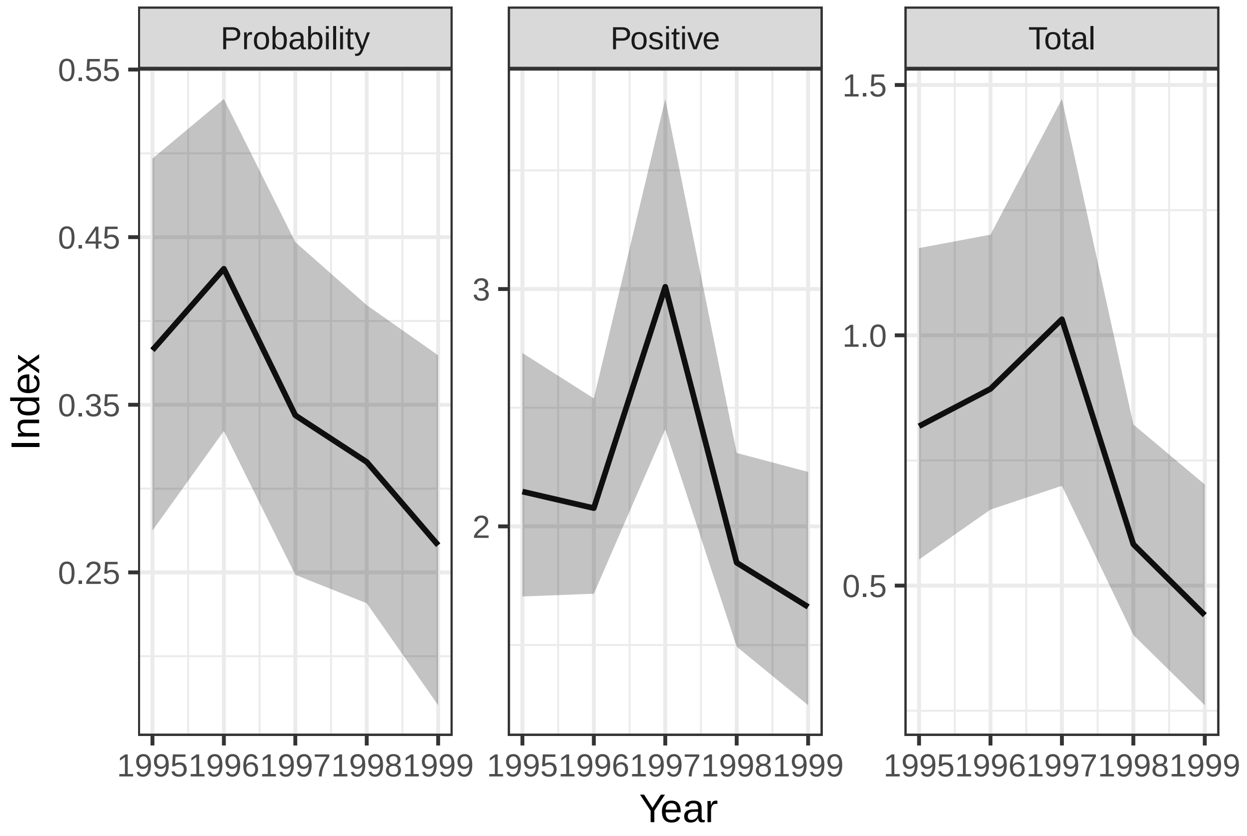
```
range(HW9_deltaLnorm_jags$BUGSoutput$summary[,"Rhat"])
```

```
## [1] 1.000988 1.001115
```

```
#summary

HW9_deltaLnorm_jags$BUGSoutput$summary
```

```
##                     mean         sd         2.5%         25%         50%
## a[1]          -0.4819409 0.24384542  -0.96959705  -0.6442266  -0.4785324
## a[2]          -0.2784829 0.20859589  -0.68849416  -0.4180404  -0.2772025
## a[3]          -0.6499916 0.22744088  -1.10637760  -0.8007442  -0.6467980
## a[4]          -0.7750823 0.21223311  -1.19932575  -0.9156936  -0.7727290
## a[5]          -1.0199120 0.27770364  -1.58018642  -1.2020303  -1.0137210
## a2[1]          0.5716638 0.11790194   0.34030848   0.4925667   0.5719047
## a2[2]          0.5385960 0.09716231   0.34831723   0.4730089   0.5387072
## a2[3]          0.9100508 0.11373704   0.68565933   0.8338564   0.9103002
## a2[4]          0.4214398 0.10851918   0.20817726   0.3484530   0.4217576
## a2[5]          0.3148609 0.14628528   0.02754184   0.2163608   0.3152615
## deviance    1004.3907247 4.76411004 997.09382207 1000.9170517 1003.7181560
## mean.lnorm[1]  2.1645534 0.26214826   1.70462032   1.9804420   2.1466338
## mean.lnorm[2]  2.0894660 0.21007301   1.71627943   1.9418149   2.0769179
## mean.lnorm[3]  3.0347168 0.35498198   2.40914620   2.7879416   3.0094125
## mean.lnorm[4]  1.8606725 0.20821987   1.49318971   1.7152723   1.8469332
## mean.lnorm[5]  1.6806233 0.25117673   1.24653735   1.5041011   1.6608182
## p.year[1]      0.3834248 0.05679241   0.27496083   0.3442917   0.3825987
## p.year[2]        0.4315534 0.05062312   0.33436814    0.3969857    0.4311397
## p.year[3]        0.3447954 0.05070547   0.24854683    0.3098663    0.3437115
## p.year[4]        0.3171525 0.04544409   0.23159518    0.2858362    0.3158891
## p.year[5]        0.2685009 0.05350617   0.17076908    0.2311142    0.2662523
## total.mean[1]    0.8298769 0.15932262   0.55210592    0.7172403    0.8184047
## total.mean[2]    0.9017928 0.14029743   0.65181732    0.8041099    0.8929603
## total.mean[3]    1.0463678 0.19752208   0.69931331    0.9082506    1.0320966
## total.mean[4]    0.5900852 0.10751507   0.40136644    0.5149533    0.5825300
## total.mean[5]    0.4513093 0.11337178   0.26107567    0.3704026    0.4410080
##                        75%        97.5%     Rhat n.eff
## a[1]            -0.3170453   -0.01247145 1.000990 95000
## a[2]            -0.1373711    0.12983833 1.000988 95000
## a[3]            -0.4949612   -0.21276399 1.000993 95000
## a[4]            -0.6313588   -0.36621462 1.001012 62000
## a[5]            -0.8300678   -0.49170479 1.001000 95000
## a2[1]            0.6507338    0.80256902 1.001100 13000
## a2[2]            0.6040138    0.72885297 1.000988 95000
## a2[3]            0.9859617    1.13437661 1.000989 95000
## a2[4]            0.4942592    0.63442351 1.000996 95000
## a2[5]            0.4130221    0.60096725 1.000997 95000
## deviance      1007.1379380 1015.43507278 1.000990 95000
## mean.lnorm[1]    2.3278374    2.73070588 1.001115 12000
## mean.lnorm[2]    2.2220552    2.53915495 1.000988 95000
## mean.lnorm[3]    3.2552786    3.80061216 1.000989 95000
## mean.lnorm[4]    1.9901140    2.30944159 1.000998 95000
## mean.lnorm[5]    1.8345820    2.22945350 1.000999 95000
## p.year[1]        0.4213960    0.49688218 1.000991 95000
## p.year[2]        0.4657111    0.53241406 1.000988 95000
## p.year[3]        0.3787255    0.44700875 1.000994 95000
## p.year[4]        0.3472025    0.40945602 1.001014 57000
## p.year[5]        0.3036307    0.37949205 1.001001 95000
## total.mean[1]    0.9299671    1.17420534 1.001019 48000
## total.mean[2]    0.9904314    1.20095943 1.000988 95000
## total.mean[3]    1.1707363    1.47198216 1.000995 95000
## total.mean[4]    0.6573584    0.82185223 1.001024 42000
## total.mean[5]    0.5203178    0.70197653 1.001012 63000
```

```r
#plot
model.bugsoutput <- HW9_deltaLnorm_jags$BUGSoutput
Nyear= marlin.list$Nyear
rows.p1<-paste0("p.year[",1:Nyear,"]")
rows.ln1<-paste0("mean.lnorm[",1:Nyear,"]")
rows.tot1<-paste0("total.mean[",1:Nyear,"]")
df1<-data.frame(model.bugsoutput$summary[rows.p1,c("2.5%","50%","97.5%")])
df3<-data.frame(model.bugsoutput$summary[rows.ln1,c("2.5%","50%","97.5%")])
df5<-data.frame(model.bugsoutput$summary[rows.tot1,c("2.5%","50%","97.5%")])
resdf1<-bind_rows(list(Probability=df1,Positive=df3,Total=df5),.id = "Type")%>%
  mutate(Type=factor(Type,levels=c("Probability","Positive","Total")))
names(resdf1)[2:4]<-c(c("min","med","max"))
resdf1$Year<-rep(sort(unique(marlin.data$Year)),3)

#Plot
ggplot(resdf1,aes(x=Year,y=med,ymin=min,ymax=max))+
  geom_line(linewidth=1)+
  geom_ribbon(alpha=0.3)+
  facet_wrap(Type~.,scale="free")+
  ylab("Index")
```



C) Which model do you think is better, a or b? Explain why you can't compare them with information criteria. Did you get the same trend across years? It is hard to tell which model is better, they both converged, though it took many more iteration for the ZIP to converge, and they both gave similar results. We can't compare the models using information criteria because the delta log normal has multiple likelihoods where as the ZIP only has one likelihood. If I had to choose, then I'd choose the delta lognormal just because it converged quicker.