SFS 2024 Short Course – Bayesian Applications in Environmental and Ecological Studies with R and Stan

Song S. Qian and Mark R. DuFour

6/1/2024

Example 3 — Juvenile Atlantic Sturgeon Relative Abundance Index

Statistical method – Zero-inflated negative binomial with covariates and hierarchical structure

In this example we expand the zero-inflated negative binomial model by incorporating covariates to explain the difference between zero's (true and false), improving parameter identifiability. In addition, we incorporate a hierarchical structure assuming exchangeability across years to evaluate abundance through time.

Background

We were interested in the relative abundance of Hudson River juvenile Atlantic sturgeon through time (2004-2015) as an index of recruitment and population recovery. Young sturgeon reside in the brackish estuarine waters of the Hudson for several years before migrating back out to sea representing a consistently targetable portion of the population. A standardized gill net survey within a fixed portion of the river during spring months (March and April) was used to monitor abundance trends. Each monitoring event consisted of a count variable (i.e., number of fish captured) as well as an exposure variable (i.e., effort = fishing time $\times$ net area). Additionally, environmental data (e.g., bottom temperature and salinity) and GPS location were recorded for each event.

The sampling environment was dynamic, and likely influenced encounter and catch rates. For example, salinity levels fluctuated within the targeted sample area as the salt-front (interface between fresh and salt water) migrated up and down river depending on river conditions such as tidal stage, discharge, and wind intensity and direction. Also, the survey began in March as water temperatures were warming, and warming rates were inconsistent across years. As a result, low abundance and variable habitat conditions generated an over-dispersed count data set with a high proportion of 0's, while the source of zero-inflation remained unclear. Data were provided by the New York State Department of Environmental Conservation - Hudson and Delaware Marine Fisheries Unit.

```r
## Importing data
zipdata <- read.csv(paste(dataDIR, "sdata_15_sub.csv", sep = "/"), header = T, sep = ",")  ## survey da

## order day by sample year
zipdata <- zipdata[order(zipdata$YEAR), ]
```

Incorporating covariates

Building on the zero-inflated negative binomial model from Example 2, we first incorporated covariates into the zero and count portions. The natural link functions for the mean (μ) and the probability of a true zero (θ) were used. In the Stan code, we can either define the negative binomial mean (i.e., $\mu = e^{\boldsymbol{X}\beta}$, using Stan sampling function `neg_binomial_2_lpmf(mu, r)` or define the log mean, `eta` (i.e., $\eta = \log(\mu)$), using the sampling function `neg_binomial_2_log_lpmf(eta, r)`. Similarly, θ can be defined as `zi` (i.e., $zi = logit(\theta)$) on the logit scale using `bernoulli_logit_lpmf(1 \| zi)` for true zero and `bernoulli_logit_lpmf(0 \| zi)` for false zero.

Following Lambert (1992) the negative binomial parameter μ and the binomial parameter θ are modeled as linear functions of predictors through the respective link function:

$$\begin{aligned} \log(\mu) &= \mathbf{X}\beta \\ \operatorname{logit}(\theta) &= \mathbf{Z}\alpha \end{aligned}$$

In this example, we used temperature as a predictor (Z) for $logit(\theta)$, and temperature and distance to salt front (X; a metric derived form the sample location and predicted salt front location) for $log(\mu)$. Exploratory data analysis suggested the relationship between distance to salt front and catch was dome shaped, so we also included log(distance to salt front) as a predictor. All covariates were centered to aid model convergence. Finally, we added effort (i.e., fishing time $\times$ net area) as an offset to represent CPUE rather than catch.

The ZINB model with covariates is :

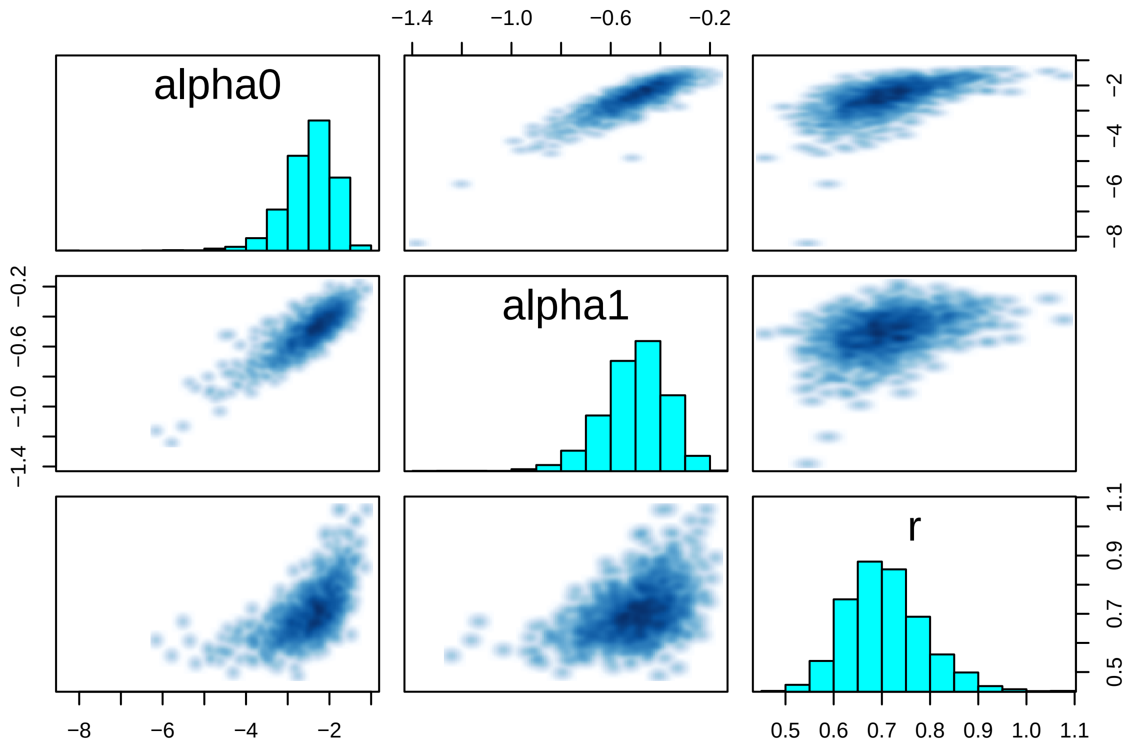
```
## Zero-inflated negative binomial stan model with covariates
zinb_sturg1 <- "
data {
  int<lower=1> N0;      // number of zero counts
  int<lower=1> Np;      // number of positive counts
  int<lower=0> Yp[Np];  // response variable (positive counts only)

  real Zp[Np];          // zero model design matrix
  real Z0[N0];          // zero model design matrix

  int<lower=1> Ka;      // number of count model coefficients
  matrix[Np, Ka] Xp;    // count model design matrix
  matrix[N0, Ka] X0;    // count model design matrix

  vector[Np] offsetP;   // exposure variable (net-soaking time)
  vector[N0] offset0;   // exposure variable (net-soaking time)
  }

parameters {
  real alpha0;          // zero model intercept
  real alpha1;          // zero model coefficient
  real b0;              // count model intercept
  vector[Ka] beta;      // count model coefficients
  real<lower=0> r;      // inverse disperson parameter
  }

transformed parameters {
  vector[Np] zi_p;          // initialize positive count predictor term zi
  vector[N0] zi_0;          // initialize zero predictor term for zi
  vector[Np] eta_p;         // initialize positive count predictor term for eta
  vector[N0] eta_0;         // initialize zero count predictor term for eta

  for (i in 1:Np){          // linear predictor for postivie counts
    eta_p[i] = b0 + Xp[i] * beta + offsetP[i];
    zi_p[i]  = alpha0 + Zp[i] * alpha1;
  }

  for (i in 1:N0){          // linear predictor for true zeroes
    eta_0[i] = b0 + X0[i] * beta + offset0[i];
    zi_0[i]  = alpha0 + Z0[i] * alpha1;
  }
  }

model {
  // priors
  alpha0 ~ normal(0,3);         // zero model intercept
  alpha1 ~ normal(0,3);         // zero model coefficient
  b0 ~ normal(0,3);             // count model intercept
  beta ~ normal(0,3);           // count model coefficients
  r ~ normal(0,2);              // inverse disperson parameter


  // likelihood
  for(i in 1:N0){
    target += log_sum_exp(bernoulli_logit_lpmf(1 | zi_0[i]),
                          bernoulli_logit_lpmf(0 | zi_0[i]) +
                          neg_binomial_2_log_lpmf(0 | eta_0[i], r));
  }

  for(i in 1:Np){
    target += bernoulli_logit_lpmf(0 | zi_p[i]) +
              neg_binomial_2_log_lpmf(Yp[i] | eta_p[i], r);
    }


}



"

################################################
fit_zinb_sturg1 <- stan_model(model_code = zinb_sturg1)
```

There are two major differences between this model and Example 2:
- θ is transformed to the logit scale zi using the `bernoulli_logit_lpmf(y\|zi)` function in Stan
- linear predictor functions are added to the transformed parameters block for `eta` and `zi`

The next set of code packages the data, generates initial values, and runs the Stan model.

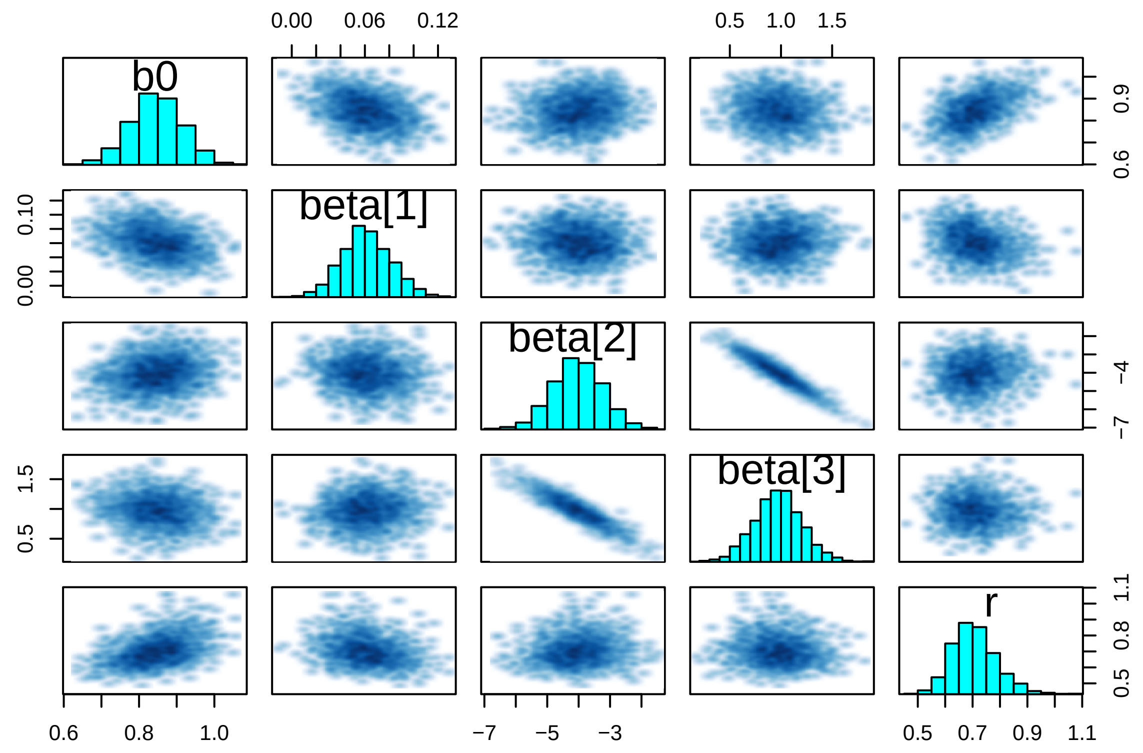
```
## bundle data, initial values, n.chains, and parameters to monitor
ZINB_in <- function(data = zipdata, n.chains = nchains) {
    y <- data$CATCH
    tmp <- !is.na(y)
    data <- data[tmp, ]
    N <- dim(data)[1]
    Y <- y[tmp]
    X <- cbind(data$TEMP - mean(data$TEMP), data$DTSF2 - mean(data$DTSF2), log(data$DTSF2) -
        mean(log(data$DTSF2)))
    Kbeta <- dim(X)[2]
    Z <- data$TEMP - mean(data$TEMP)
    offsets <- log(data$Effort)
    tmp <- y == 0
    np <- sum(!tmp)
    n0 <- sum(tmp)
    data <- list(Np = np, N0 = n0, Ka = Kbeta, Xp = X[!tmp, ], X0 = X[tmp, ], Yp = y[!tmp],
        Zp = Z[!tmp], Z0 = Z[tmp], offset0 = offsets[tmp], offsetP = offsets[!tmp])  #,
    inits <- list()
    for (i in 1:n.chains) inits[[i]] <- list(alpha0 = rnorm(1), alpha1 = rnorm(1),
        b0 = rnorm(1), beta = rnorm(Kbeta), r = (runif(1)))
    paras <- c("alpha0", "alpha1", "b0", "beta", "r")
    return(list(data = data, init = inits, nchains = n.chains, para = paras))
}

input.to.stan <- ZINB_in()

## run model
keep_zinb_sturg1 <- sampling(fit_zinb_sturg1, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchain)

## save(keep_zinb_sturg1,file='zinb_sturg1.RData')

## view pairs plots
pairs(keep_zinb_sturg1, pars = c("alpha0", "alpha1", "r"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



```r
pairs(keep_zinb_sturg1, pars = c("b0", "beta", "r"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



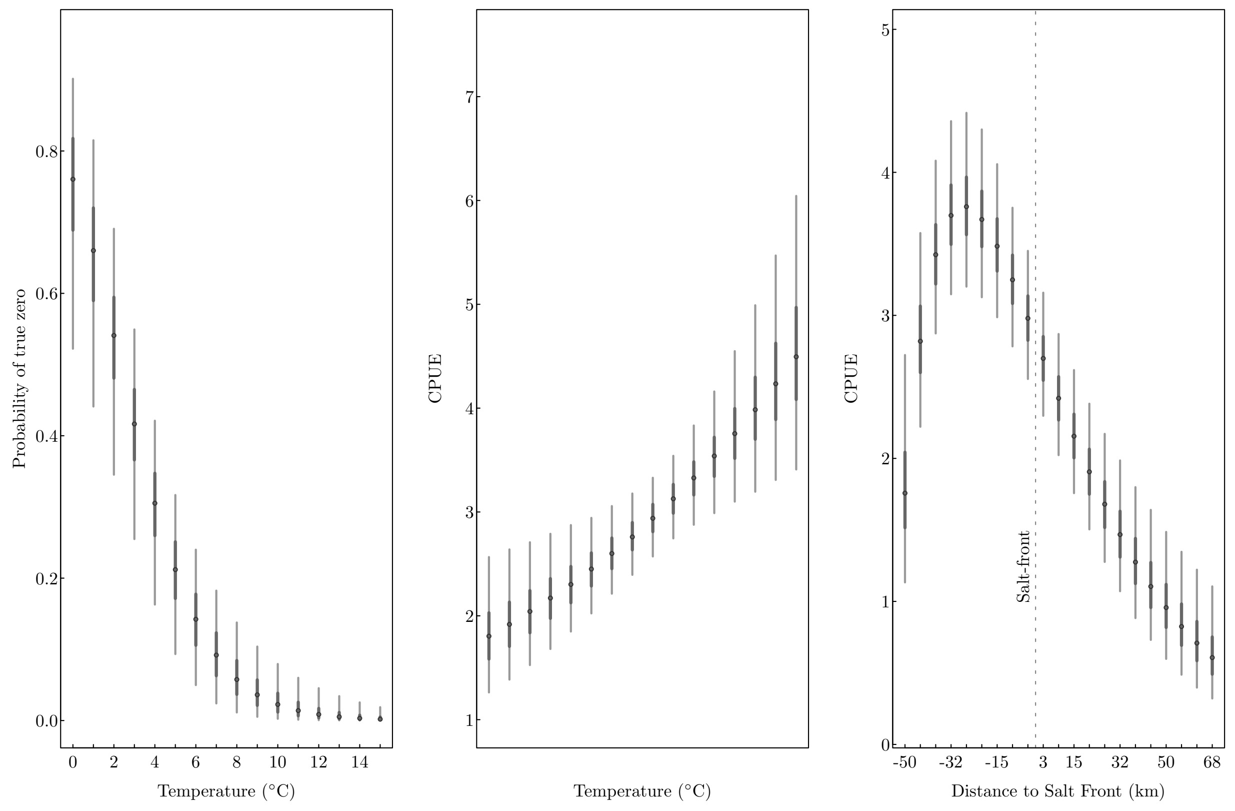
The model runs quickly and does not produce any warnings. When we examine the pairs plot, there is strong correlation between the beta[1] and beta[2] parameters which we'd expect (part of gamma model), but no other concerning patterns (e.g., Neal's funnel) occurring. In the zero portion of the model, we still see banana shaped correlation patterns between r and alpha0 and alpha1 indicating convergence trouble. However, the pattern is not nearly as pronounced as we saw in Example 2 suggesting the addition of covariates has helped.

What have the covariates told us about the sturgeon catch data?

```r
## extract MCMC draws and summarize parameters
fitcoef <- rvsims(as.matrix(as.data.frame(extract(keep_zinb_sturg1, permute = T))))

options(digits = 3)
summary(fitcoef[1:7])
```

```
##        name    mean     sd      1%    2.5%    25%     50%     75%  97.5%    99%
## [1] alpha0 -2.4829 0.6114 -4.3750 -3.8778 -2.825 -2.3967 -2.0553 -1.573 -1.465
## [2] alpha1 -0.5030 0.1301 -0.8623 -0.7940 -0.581 -0.4908 -0.4114 -0.288 -0.250
## [3]     b0  0.8453 0.0698  0.6856  0.7041  0.799  0.8452  0.8922  0.982  1.001
## [4] beta.1  0.0615 0.0202  0.0163  0.0223  0.048  0.0607  0.0749  0.102  0.108
## [5] beta.2 -4.0268 0.8120 -6.0268 -5.6785 -4.567 -4.0372 -3.4856 -2.476 -2.213
## [6] beta.3  0.9728 0.2363  0.4154  0.5257  0.816  0.9735  1.1200  1.449  1.525
## [7]      r  0.7067 0.0819  0.5444  0.5658  0.649  0.6989  0.7545  0.886  0.925
##     sims
## [1] 4000
## [2] 4000
## [3] 4000
## [4] 4000
## [5] 4000
## [6] 4000
## [7] 4000
```

```
## un-center intercepts
real_zi_intercept <- fitcoef[1] - (fitcoef[2] * mean(zipdata$TEMP))

real_Intercept <- fitcoef[3] - (fitcoef[4] * mean(zipdata$TEMP) + fitcoef[5] * mean(zipdata$DTSF2) +
    fitcoef[6] * mean(log(zipdata$DTSF2)))



## plot model estimated relationships tikz(file=paste(plotDIRch5,
## 'zinb_sturg_coefs.tex', sep='/'), height=2, width=4.75, standAlone=F)
par(mfrow = c(1, 3), mar = c(3, 3, 3, 1), mgp = c(1.5, 0.125, 0), las = 1, tck = 0.01)

## Temp effect on zi
tmp <- substring(names(fitcoef), 1, 1) == "a"
zero_coef <- fitcoef[tmp]
x.tmp <- seq(0, 15, by = 1)
tmp.zi <- real_zi_intercept + zero_coef[2] * x.tmp
plot((invlogit(tmp.zi)), pch = 1, xaxt = "n", cex = 0.5, xlab = "Temperature ($^{\\circ}$C)",
    ylab = "Probability of true zero")
axis(1, at = c(1:16), labels = x.tmp)


## DTSF effect on counts
int <- substring(names(fitcoef), 1, 2) == "b0"
bets <- fitcoef[substring(names(fitcoef), 1, 4) == "beta"]
dtsf <- zipdata$DTSF2
tmp.c <- real_Intercept + bets[1] * x.tmp + bets[2] * mean(zipdata$DTSF2) + bets[3] *
    mean(log(zipdata$DTSF2)) + mean(log(zipdata$Effort))
plot(exp(tmp.c), xaxt = "n", pch = 1, cex = 0.5, xlab = "Temperature ($^{\\circ}$C)",
    ylab = "CPUE")

x.dtsf <- seq(0.05, 1.05, by = 0.05)
dtsf.c <- real_Intercept + bets[1] * mean(zipdata$TEMP) + bets[2] * x.dtsf + bets[3] *
    log(x.dtsf) + mean(log(zipdata$Effort))
plot(exp(dtsf.c), xaxt = "n", pch = 1, cex = 0.5, xlab = "Distance to Salt Front (km)",
    ylab = "CPUE")
axis(1, at = c(1:21), labels = round(seq(min(zipdata$DTSF), max(zipdata$DTSF), length.out = 21),
    0))
abline(v = 9.5, lty = 3, col = gray(0, 0.5))
text(9.5, 1.5, "Salt-front", pos = 2, srt = 90)
```



```
# dev.off()
```

Probability of a true zero declines precipitously as water temperatures increase, likely due to increased fish movement and encounter rates with gill nets. CPUE increases with water temperature as fish move into the sample area, and the highest CPUE occurred when the sample area was in brackish waters below the salt front.

Ultimately, we were interested in evaluating the change in abundance over time. Next, we build on the current model by incorporating a year specific intercept.

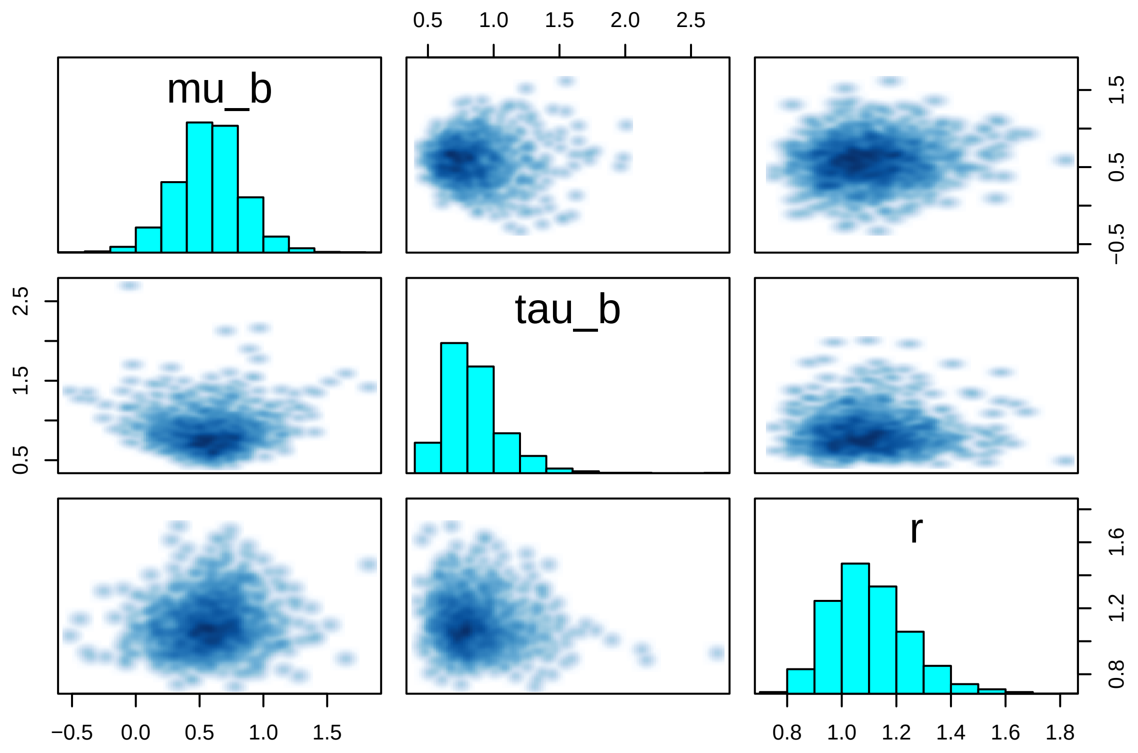
```
## Zero-inflated negative binomial stan model with covariates and year
## hierarchy

zinb_sturg2 <- "
data {
  int<lower=1> N0;      // number of zero counts
  int<lower=1> Np;      // number of positive counts
  int<lower=0> Yp[Np];  // response variable (+ counts only)

  real Zp[Np];          // zero model design matrix
  real Z0[N0];          // zero model design matrix

  int<lower=1> Ka;      // number of count model coefficients
  matrix[Np, Ka] Xp;    // count model design matrix
  matrix[N0, Ka] X0;    // count model design matrix

  vector[Np] offsetP;   // exposure variable (net-soaking time)
  vector[N0] offset0;   // exposure variable (net-soaking time)

  int<lower=1> Nyr;                    // number of years
  int<lower=1,upper=Nyr> yearP[Np];   // year hierarchy
  int<lower=1,upper=Nyr> year0[N0];   // year hierarchy
  }

parameters {
  real alpha0;          // zero model intercept
  real alpha1;          // zero model coefficient
  vector[Ka] beta;      // count model coefficients
  real<lower=0> r;      // inverse disperson parameter

  vector[Nyr] b0;       // count model year specific intercept
  real mu_b;            // hyper mean for b0
  real<lower=0> tau_b;  // hyper sd for b0
  }

transformed parameters{
  vector[Np] zi_p;          // initialize postive count predictor term for zi
  vector[N0] zi_0;          // initialize zero predictor term for zi
  vector[Np] eta_p;         // initialize positive count predictor term for eta
  vector[N0] eta_0;         // initialize zero count predictor term for eta

  for (i in 1:Np){          // linear predictor for postivie counts
    eta_p[i] = b0[yearP[i]]+ Xp[i] * beta + offsetP[i];
    zi_p[i]  = alpha0 + Zp[i] * alpha1;
  }

  for (i in 1:N0){          // linear predictor for true zeros
    eta_0[i] = b0[year0[i]]+ X0[i] * beta + offset0[i];
    zi_0[i]  = alpha0 + Z0[i] * alpha1;
  }

  }

model {
  // priors
  alpha0 ~ normal(0,5);         // zero model intercept
  alpha1 ~ normal(0,5);         // zero model coefficients
  beta ~ normal(0,5);           // count model coefficients
  r ~ gamma(0.001, 0.001);        // inverse dispersion parameter

  b0 ~ normal(mu_b, tau_b);     // year specific count model intercepts
  mu_b ~ normal(0,5);           // hyper mean for b0
  tau_b ~ normal(0,5);          // hyper sd for b0

  // likelihood
  for(i in 1:N0){
    target += log_sum_exp(bernoulli_logit_lpmf(1 | zi_0[i]),
                       bernoulli_logit_lpmf(0 | zi_0[i]) +
                       neg_binomial_2_log_lpmf(0 | eta_0[i], r));
    }
  for(i in 1:Np){
    target += bernoulli_logit_lpmf(0 | zi_p[i]) +
              neg_binomial_2_log_lpmf(Yp[i] | eta_p[i], r);
    }
}
"

###############################################
fit_zinb_sturg2 <- stan_model(model_code = zinb_sturg2)
```

There are two major differences from between `fit_zinb_sturg2` and the `fit_zinb_sturg1` models:
- The `b0` intercept in the positive count model is allowed to vary by year (`b0[year0[i]]` and `b0[yearP[i]]`)
- All `b0` now have a common mean (`mu_b`) and standard deviation (`tau_b`) prior

The next set of code packages the data, generates initial values, and runs the Stan model.

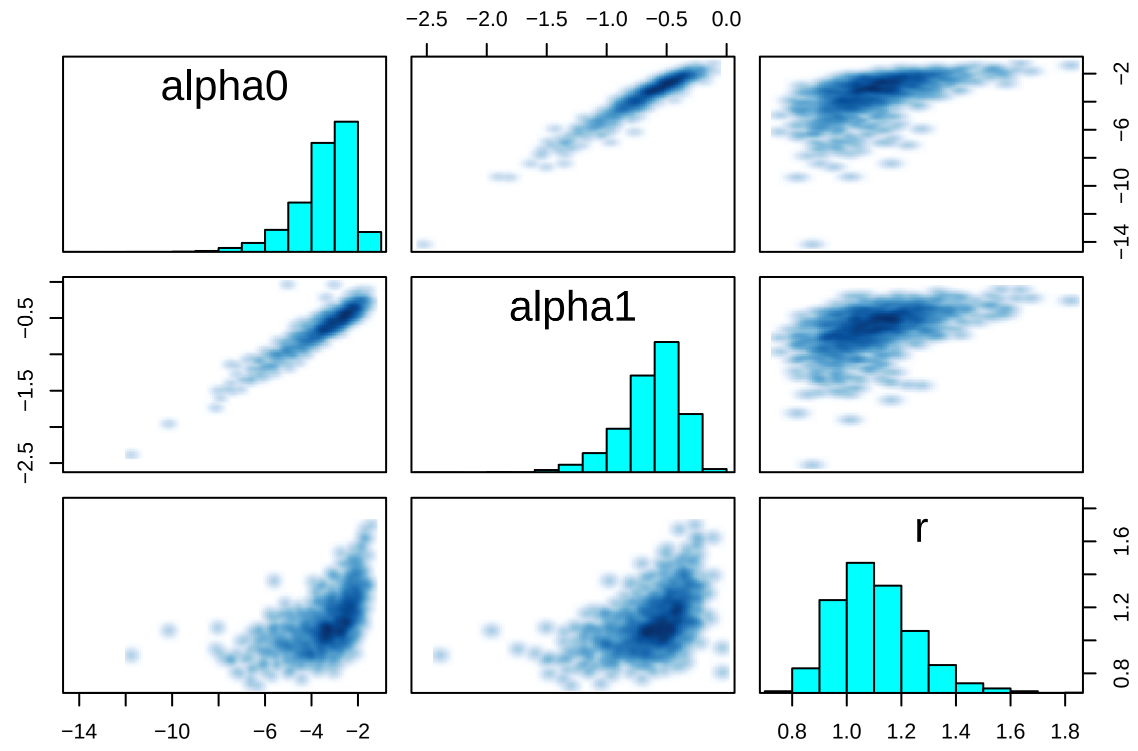
```
## bundle data, initial values, n.chains, and parameters to monitor
ZINB_in <- function(data = zipdata, n.chains = nchains) {
    y <- data$CATCH
    tmp <- !is.na(y)
    data <- data[tmp, ]
    N <- dim(data)[1]
    Y <- y[tmp]
    X <- cbind(data$TEMP - mean(data$TEMP), data$DTSF2 - mean(data$DTSF2), log(data$DTSF2) -
        mean(log(data$DTSF2)))
    Kbeta <- dim(X)[2]
    Z <- data$TEMP - mean(data$TEMP)
    offsets <- log(data$Effort)
    year <- as.numeric(ordered(data$YEAR))
    nyr <- max(year)
    tmp <- y == 0
    np <- sum(!tmp)
    n0 <- sum(tmp)
    data <- list(Np = np, N0 = n0, Nyr = nyr, Ka = Kbeta, Xp = X[!tmp, ], X0 = X[tmp,
        ], Yp = y[!tmp], Zp = Z[!tmp], Z0 = Z[tmp], offset0 = offsets[tmp], offsetP = offsets[!tmp],
        yearP = year[!tmp], year0 = year[tmp])
    inits <- list()
    for (i in 1:n.chains) inits[[i]] <- list(alpha0 = rnorm(1), alpha1 = rnorm(1),
        b0 = rnorm(nyr), beta = rnorm(Kbeta), r = runif(1), tau_b = runif(1))  # mu_b=rnorm(1),
    paras <- c("alpha0", "alpha1", "b0", "beta", "r", "tau_b", "mu_b")
    return(list(data = data, init = inits, nchains = n.chains, para = paras))
}

input.to.stan <- ZINB_in()

## fun model
keep_zinb_sturg2 <- sampling(fit_zinb_sturg2, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchains)

## save(keep_zinb_sturg2,file='zinb_sturg2.RData')

## view pairs plots
pairs(keep_zinb_sturg2, pars = c("mu_b", "tau_b", "r"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



```
# pairs(keep_zinb_sturg2, pars = c('b0','beta','r'))
pairs(keep_zinb_sturg2, pars = c("alpha0", "alpha1", "r"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



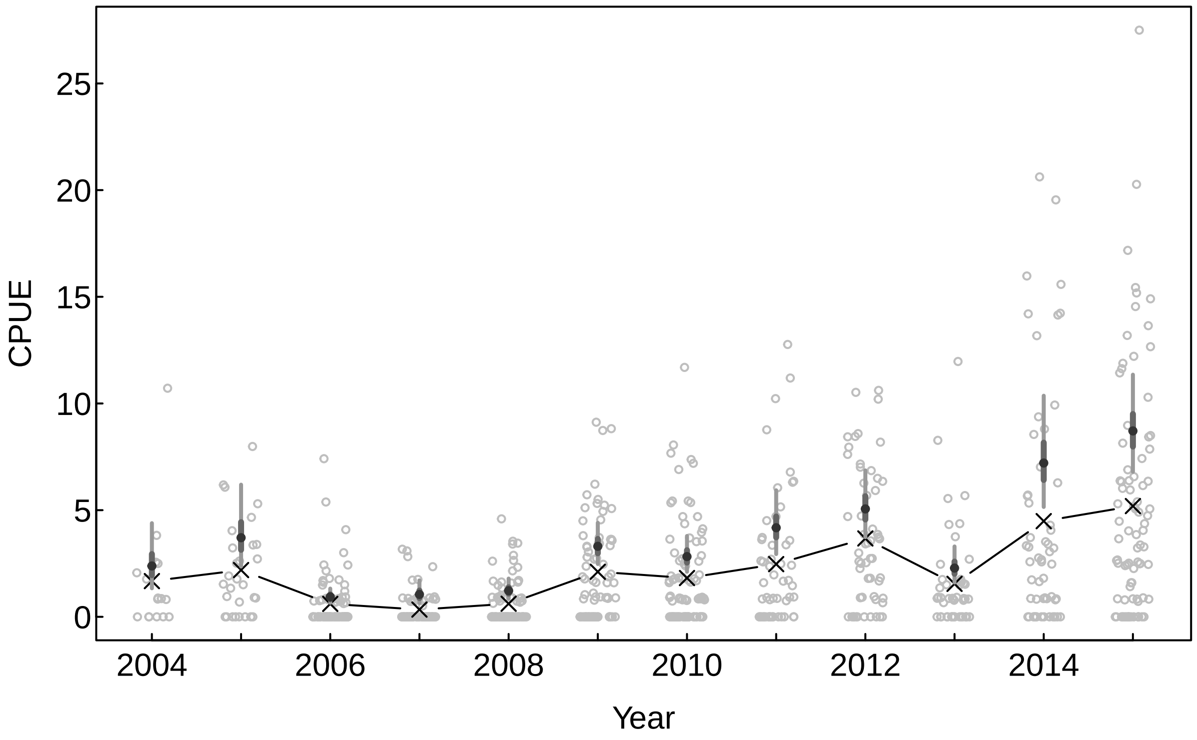
The model runs quickly without producing any warnings. Looking over the pair plots, there are no concerning patterns in the upper level parameters. Correlations between `r` and the `b0` and `beta[1:12]` look good. However, the non-identifiability issue in the zero model parameters (`alpha0` and `alpha1`) remains.

Estimating year specific intercepts in the count model allows us to control for the covariates which influence survey performance, and estimate relative abundance under ideal sampling conditions.

```
## extract MCMC draws and summarize parameters
fitcoef <- rvsims(as.matrix(as.data.frame(extract(keep_zinb_sturg2, permute = T))))

options(digits = 3)
summary(fitcoef[1:20])
```

```
##         name    mean     sd       1%     2.5%     25%     50%     75%   97.5%
##   [1] alpha0 -3.4004 1.2334 -7.39960 -6.48759 -3.9811 -3.1416 -2.5241 -1.8196
##   [2] alpha1 -0.6244 0.2553 -1.41548 -1.24869 -0.7514 -0.5817 -0.4477 -0.2518
##   [3]   b0.1  0.4037 0.3021 -0.28273 -0.16695  0.2011  0.3952  0.5985  1.0018
##   [4]   b0.2  0.8454 0.2355  0.30751  0.41006  0.6822  0.8391  1.0137  1.3118
##   [5]   b0.3 -0.5406 0.1864 -0.96764 -0.90434 -0.6651 -0.5407 -0.4148 -0.1782
##   [6]   b0.4 -0.4275 0.2293 -0.95173 -0.85544 -0.5879 -0.4257 -0.2753  0.0314
##   [7]   b0.5 -0.2680 0.1711 -0.65344 -0.59669 -0.3851 -0.2706 -0.1559  0.0710
##   [8]   b0.6  0.7221 0.1438  0.40030  0.44355  0.6213  0.7204  0.8190  1.0074
##   [9]   b0.7  0.5642 0.1394  0.23791  0.28669  0.4714  0.5623  0.6575  0.8346
##  [10]   b0.8  0.9600 0.1692  0.58128  0.62403  0.8485  0.9565  1.0690  1.2997
##  [11]   b0.9  1.1515 0.1632  0.79100  0.83340  1.0390  1.1486  1.2637  1.4863
##  [12]  b0.10  0.3559 0.1832 -0.07935  0.00486  0.2340  0.3494  0.4757  0.7162
##  [13]  b0.11  1.5077 0.1571  1.14372  1.21102  1.4027  1.5048  1.6099  1.8222
## [14]   b0.12  1.6938 0.1241  1.40189  1.44660  1.6095  1.6948  1.7749  1.9378
## [15]  beta.1  0.0524 0.0204  0.00407  0.01133  0.0386  0.0528  0.0663  0.0916
## [16]  beta.2 -3.6458 0.7456 -5.39782 -5.14845 -4.1249 -3.6389 -3.1288 -2.2345
## [17]  beta.3  0.8015 0.2151  0.29702  0.38611  0.6524  0.7976  0.9468  1.2182
## [18]       r  1.1034 0.1476  0.82307  0.85548  0.9977  1.0864  1.1879  1.4293
## [19]   tau_b  0.8465 0.2213  0.49566  0.52979  0.6941  0.8078  0.9549  1.3813
## [20]    mu_b  0.5763 0.2631 -0.07753  0.05493  0.4117  0.5823  0.7405  1.1095
##           99% sims
##  [1] -1.6770 4000
##  [2] -0.2121 4000
##  [3]  1.1390 4000
##  [4]  1.3920 4000
##  [5] -0.0981 4000
##  [6]  0.1154 4000
##  [7]  0.1347 4000
##  [8]  1.0654 4000
##  [9]  0.8875 4000
## [10]  1.3628 4000
## [11]  1.5428 4000
## [12]  0.7703 4000
## [13]  1.8916 4000
## [14]  1.9920 4000
## [15]  0.0988 4000
## [16] -1.9888 4000
## [17]  1.3183 4000
## [18]  1.5359 4000
## [19]  1.5470 4000
## [20]  1.2492 4000
```

```
## un-center intercepts
real_zi_intercept <- fitcoef[1] - (fitcoef[2] * mean(zipdata$TEMP))
real_Intercept_12 <- fitcoef[3:14] - (fitcoef[15] * mean(zipdata$TEMP) + fitcoef[16] *
    mean(zipdata$DTSF2) + fitcoef[17] * mean(log(zipdata$DTSF2)))

## standardized index - fishing at survey mean temp and always below saltfront
## near abundance peak
std_index <- exp(real_Intercept_12 + fitcoef[15] * mean(zipdata$TEMP) + fitcoef[16] *
    0.2 + fitcoef[17] * log(0.2) + mean(log(zipdata$Effort)))


# tikz(file=paste(plotDIRch5, 'zinbPred.tex', sep='/'), height=2.75,
# width=3.25, standAlone=F)
par(mfrow = c(1, 1), mar = c(3, 3, 3, 1), mgp = c(1.5, 0.125, 0), las = 1, tck = 0.01)
x.dtsf <- seq(0, 1, by = 0.01)
plot(jitter(as.numeric(as.factor(zipdata$YEAR))), zipdata$CATCH/zipdata$Effort, pch = 1,
    col = "gray", cex = 0.5, xaxt = "n", xlab = "Year", ylab = "CPUE")

nom_index <- aggregate(CATCH/Effort ~ YEAR, data = zipdata, FUN = "mean")
points(nom_index$"CATCH/Effort", col = "black", typ = "b", pch = 4)

points((std_index), cex = 0.5)
axis(1, at = c(1:12), labels = c(2004:2015))
```



```r
# dev.off()
```

Using the count portion of the model, we estimated CPUE under two conditions: 1) always fishing at the survey mean temp, and 2) always fishing below the salt front where CPUE peaks. This yielded a standardized index with higher CPUE in general, and substantial increases in year like 2014 and 2015 that ether sampled during cool periods or primarily above the salt front.

Summary

This example expanded the zero-inflated negative binomial model by incorporating covariates and adding hierarchical structure (i.e., year specific intercepts). The addition of covariates improved parameter identifiability, but did not completely eliminate the issue. Had we understood this problem earlier, we likely would have fit the model using only a negative binomial distribution. Assuming exchangeability across year specific intercepts helped inform a management problem (i.e., Is the population growing?), making the model more useful.