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 2 – Zero-Inflation or Not?

In this example, we introduce the concept of Bayesian posterior simulation for model evaluation using a scenario often encountered in ecological data.

Incidental Catch in Fisheries example (Hilborn and Mangel, 1997. *Ecological Detective*) aimed to determine the minimum sample size of "statistically meaningful data" required to estimate the mean bycatch rate with a limited level of uncertainty. The original study (Bartle, 1991) used Central Limit Theorem-based confidence intervals to define an acceptable level of uncertainty. The goal was to estimate the mean and variance of the bycatch numbers. Hilborn and Mangel (1997) utilized the negative binomial distribution to describe the bycatch data, where the response variable is the bycatch count, a count variable taking only non-negative integer values.

Fisher et al. (1943) suggested three types of distributions for biological measurements, including the Poisson distribution for count variables.

The Poisson distribution has one parameter (λ), representing the mean. The probability function is given by

$$\pi(Y = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

When observing independent observations $y_1, \cdots, y_n$, the log-likelihood function is

$$LL = \log\left(\prod_{i=1}^{n} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}\right) \propto \log(\lambda)\sum_{i=1}^{n} y_i - n\lambda$$

In a Bayesian statistics, we normally use the gamma distribution as the prior for λ, because the posterior distribution of λ is also a gamma distribution. If the prior is $\lambda \sim \text{gamma}(\alpha, \beta)$, the posterior is $\lambda \mid y_1, \ldots, y_n \sim \text{gamma}(\alpha + \sum_{i=1}^{y}, \beta + n)$. The gamma distribution is the same as the χ^2 distribution, which Fisher et al (1943) used to derive the negative binomial distribution.

We fit the bycatch data using both Poisson and negative binomial distributions to show why the negative binomial model is more useful.

```
zipdata <- data.frame(Capture = 0:17, numb = c(807, 37, 27, 8, 4, 4, 1, 3, 1, 0,
    0, 2, 1, 1, 0, 0, 0, 1))
zippos <- rep(zipdata[-1, 1], zipdata$numb[-1])
zip0 <- zipdata$numb[1]

################################################ Simple models (without zero
################################################ inflation) ####

## the Poisson model
Pois_bycatch <- "
```

```
data{
      int<lower=1> n0; //number of 0's
      int<lower=1> np; //number of non-zero counts
      int<lower=1> yp[np];
   }
   parameters{
      real<lower=0> lambda;
   }
   model{
     target += n0*poisson_log_lpmf(0|log(lambda));
     target += poisson_log_lpmf(yp|log(lambda));
   }
"
fitPois <- stan_model(model_code = Pois_bycatch)

## Neg_binomial
NB_bycatch <- "
data{
      int<lower=1> n0; //number of 0's
      int<lower=1> np; //number of non-zero counts
      int<lower=1> yp[np];
   }
   parameters{
      real<lower=0> mu;
      real<lower=0> r;
   }
   model{
     mu ~ normal(0,2);
     r ~ normal(0,2);
     target += n0*neg_binomial_2_log_lpmf(0 | log(mu), r);
     target += neg_binomial_2_log_lpmf(yp | log(mu), r);
   }
"
fitNB <- stan_model(model_code = NB_bycatch)

save(fitNB, fitPois, file = "zip1.RData")
```

We separated 0s from non-zero observations to make the code more comparable to the zero-inflated alternatives.

```
## one input function for both models
bycatch_input <- function(yp = zippos, n0 = zip0, model = "Pois", n.chains = nchains) {
    N <- length(yp) + n0
    np <- length(yp)
    data <- list(np = np, n0 = n0, yp = yp)
    inits <- list()
    for (i in 1:n.chains) {
        if (model == "Pois")
            inits[[i]] <- list(lambda = runif(1)) else inits[[i]] <- list(mu = runif(1), r = runif(1))
    }
    if (model == "Pois")
        pars <- c("lambda") else pars <- c("mu", "r")
    return(list(data = data, init = inits, nchains = n.chains, para = pars, model = model))
}
```

Fitting the model

```r
## the Poisson model
input.to.stan <- bycatch_input()
keep_Pois <- sampling(fitPois, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchains)  ##,
## control=list(max_treedepth=20))
print(keep_Pois)
```

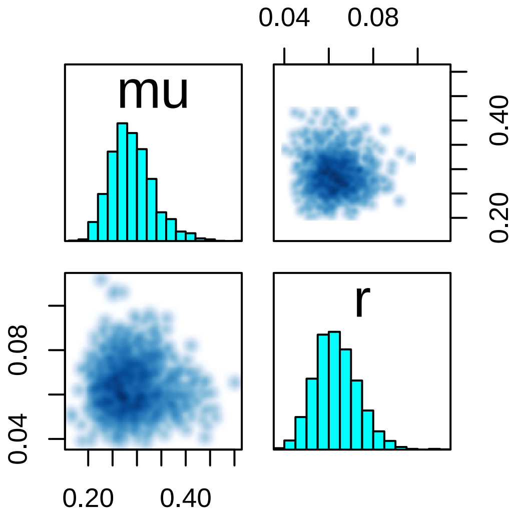
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## lambda    0.28    0.00 0.02    0.25    0.27    0.28    0.29    0.32  2482    1
## lp__   -789.90    0.01 0.73 -791.88 -790.09 -789.63 -789.44 -789.39  2383    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:00:09 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
Pois_stanout <- extract(keep_Pois, pars = "lambda")
Pois_coef <- rvsims(as.matrix(as.data.frame(Pois_stanout)))

## negative bin:
input.to.stan <- bycatch_input(model = "NB")
keep_NB <- sampling(fitNB, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchains)  ##,
## control=list(max_treedepth=20))
print(keep_NB)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## mu      0.29    0.00 0.04    0.22    0.26    0.29    0.32    0.39  2639    1
## r       0.06    0.00 0.01    0.05    0.06    0.06    0.07    0.08  2452    1
## lp__ -457.20    0.02 1.04 -460.03 -457.62 -456.86 -456.46 -456.19  2220    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:00:34 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

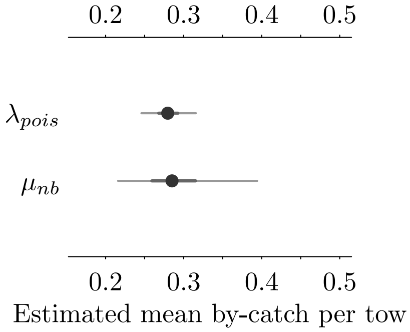
```r
pairs(keep_NB, pars = c("mu", "r"))
```

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



```
NB_stanout <- extract(keep_NB, pars = c("mu", "r"))
NB_coef <- rvsims(as.matrix(as.data.frame(NB_stanout)))
```

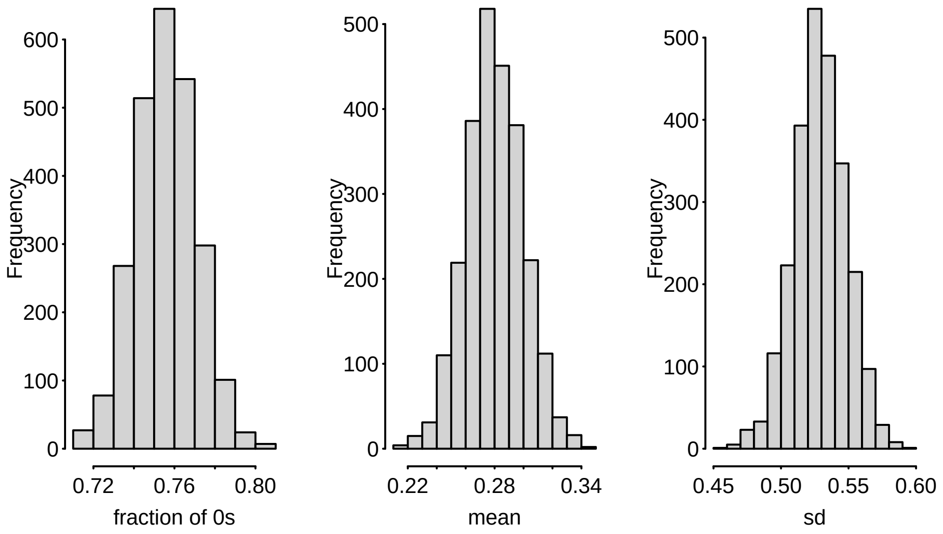
```
## Comparison
means <- c(Pois_coef[1], NB_coef[1])
names(means) <- c("$\\lambda_{pois}$", "$\\mu_{nb}$")

par(mfrow = c(1, 1), mgp = c(1.25, 0.25, 0), tck = -0.01)
mlplot(means, xlab = "Estimated mean by-catch per tow")
```



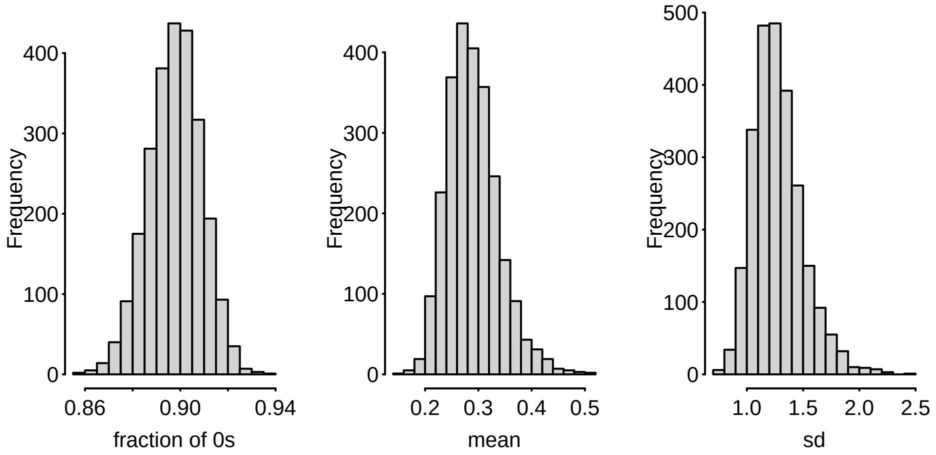
Posterior simulation – using the fitted model to replicate the data repeatedly

```
## simulations
nsims <- length(Pois_stanout$lambda)
n <- length(zippos)+zip0
```

```
### Poisson model
n0_pois <- mu_pois <- sig_pois <- NULL
for (i in 1:nsims){
    tmp <- rpois(n, lambda=Pois_stanout$lambda[i])
    n0_pois <- c(n0_pois, mean(tmp==0))
    mu_pois <- c(mu_pois, mean(tmp))
    sig_pois <- c(sig_pois, sd(tmp))
}
### here is why I like `rv`
tmp <- rvpois(1, Pois_stanout$lambda)
par(mfrow=c(1,3), mar=c(3, 3, 1, 1), mgp=c(1.5, 0.25, 0), las=1, tck=-0.01)
hist(summary((tmp==0))$mean, xlab="fraction of 0s", main="")
hist(summary(tmp)$mean, xlab="mean", main="")
hist(summary(tmp)$sd, xlab="sd", main="")
```

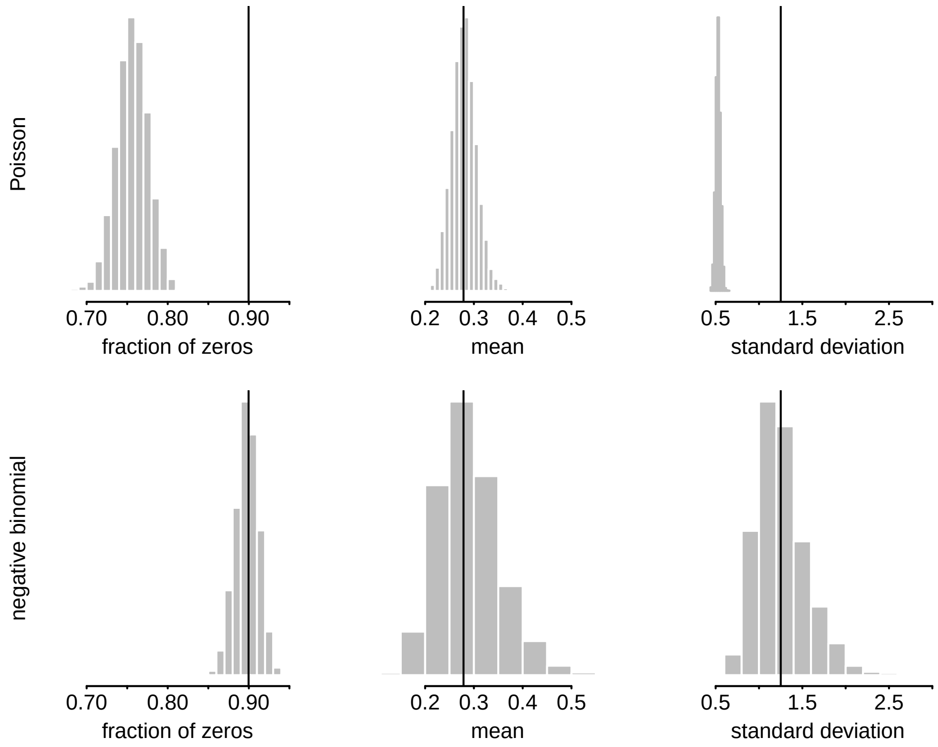


```
### negative Bin:
n0_NB <- mu_NB <- sig_NB <- NULL
for (i in 1:nsims){
    tmp <- rnbinom(n, mu=NB_stanout$mu[i], size=NB_stanout$r[i])
    n0_NB <- c(n0_NB, mean(tmp==0))
    mu_NB <- c(mu_NB, mean(tmp))
    sig_NB <- c(sig_NB, sd(tmp))
}
## or use rv (generating from gamma-poisson conjugate):
tmp1 <- rvgamma(1, NB_stanout$r, NB_stanout$r/NB_stanout$mu)
tmp <- rvpois(1, tmp1)
par(mfrow=c(1,3), mar=c(3, 3, 1, 1), mgp=c(1.5, 0.25, 0), las=1, tck=-0.01)
hist(summary(tmp==0)$mean, xlab="fraction of 0s", main="")
hist(summary(tmp)$mean, xlab="mean", main="")
hist(summary(tmp)$sd, xlab="sd", main="")
```



Put them together

```
par(mfrow=c(2,3), mar=c(2.5,2.5,1,1), mgp=c(1.25,0.125,0), tck=-0.01)
hist(n0_pois, xlim=range(c(n0_pois, n0_NB, zip0/n)),
     xlab="fraction of zeros", ylab="Poisson", main="",
     border="white", density=-1, col="gray", yaxt="n")
abline(v=zip0/n)
hist(mu_pois,
     xlim=range(c(mu_pois, mu_NB, mean(c(rep(0,zip0),zippos)))),
     xlab="mean", ylab="", yaxt="n", main="",
     border="white", density=-1, col="gray")
abline(v=mean(c(rep(0,zip0),zippos)))
#hist(sig_pois,
#     xlim=range(c(sig_pois, sig_NB, sd(c(rep(0,zip0),zippos)))),
#     xlab="standard deviation", ylab="", yaxt="n", main="",
#     border="white", density=-1, col="gray")
hist(sig_pois,
     xlim=range(c(sig_pois, sig_NB,sd(c(rep(0,zip0),zippos)))),
     xlab="standard deviation", ylab="", main="",yaxt="n",
     border="gray", density=-1, col="gray")
abline(v=sd(c(rep(0,zip0),zippos)))

hist(n0_NB, xlim=range(c(n0_pois, n0_NB,zip0/n)),yaxt="n",
     xlab="fraction of zeros", ylab="negative binomial", main="",
     border="white", density=-1, col="gray")
abline(v=zip0/n)
hist(mu_NB,
     xlim=range(c(mu_pois, mu_NB,mean(c(rep(0,zip0),zippos)))),
     xlab="mean", ylab="", main="",yaxt="n",
     border="white", density=-1, col="gray")
abline(v=mean(c(rep(0,zip0),zippos)))
hist(sig_NB,
     xlim=range(c(sig_pois, sig_NB,sd(c(rep(0,zip0),zippos)))),
     xlab="standard deviation", ylab="", main="",yaxt="n",
     border="white", density=-1, col="gray")
abline(v=sd(c(rep(0,zip0),zippos)))
```



The Poisson model cannot replicate the variance and the fraction of zeros of the data. The data are zero inflated if the Poisson model is used.

Zero-Inflated count data models

The zero-inflated model is designed to handle count data that exhibits excess zeros. Under certain conditions, the observations are always 0 (true zero). For example, there were no sea birds present. Without knowing the conditions, we must treat observed 0s as a combination of true zeros and 0s resulted from, for example, the imperfect sampling method (i.e., bycatch happened but failed to observe). Assuming that the chance of sampling a true 0 is θ (and a probability of $1-\theta$ sampling a Poisson counts). The probability of observing any outcome y_i is defined as follows: - The probability of observing 0 is given by

$$\pi(y_i) = \theta + (1-\theta) \cdot Pois(0|\lambda),$$

where $Pois(0|\lambda)$ represents the probability of observing a zero count in a Poisson distribution with mean λ. - The probability of observing $y_i > 0$ is given by

$$\pi(y_i) = (1-\theta) \cdot Pois(y_i|\lambda),$$

representing the probability of observing a non-zero count in the Poisson distribution. - Combined together:

$$\pi(y_i) = \begin{cases} \theta + (1-\theta) Pois(0 \mid \lambda) & \text{if } y_i = 0 \\ (1-\theta) Pois(y_i \mid \lambda) & \text{if } y_i > 0. \end{cases}$$

The likelihood function is

$$L = [\theta + (1-\theta)e^{-\lambda}]^{n_0} \times \prod_{i=1}^{n_p} (1-\theta) \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$$

The log-likelihood function is

$$LL = n_0 \log[\theta + (1-\theta)e^{-\lambda}] + \left(n_p \log(1-\theta) + \sum_{i=1}^{n} \log\left(\frac{\lambda^{y_i} e^{-\lambda}}{y_i!} \right) \right)$$

Writing the log-likelihood in Stan:

```
model{
  target += n0*log_sum_exp(log(theta), log1m(theta)-lambda);
  target += np*log1m(theta) + poisson_lpmf(yp| lambda);
}
```

Because $\log(\theta)$ is the log of probability of observing a true 0, we can also directly use the Stan function `bernoulli_logit_lpmf` to write the code:

```
model{
  target += n0*log_sum_exp(bernoulli_logit_lpmf(1|zi),
bernoulli_logit_lpmf(0|zi)+poisson_log_lpmf(0|eta));
  target += np*bernoulli_logit_lpmf(0|zi) +
            poisson_log_lpmf(yp | eta);
}
```

where `zi` is the logit of the probability of a true zero ($z_i = \text{logit}(\theta)$) and `eta` is the log of the Poisson parameter ($\eta = \log(\lambda)$).

Lambert (1992) introduced a generalized linear model for zero-inflated Poisson process. In the model, the Poisson model parameter λ and the binomial model parameter θ are modeled as linear functions of predictors through the respective link function:

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

The R package `pscl` implements the MLE of the Lambert ZIP model.

Fitting the models:

```
zip_bycatch <-"
data{
  int<lower=1> n0;
  int<lower=1> np;
  int<lower=1> yp[np];
}
parameters{
  real<lower=0,upper=1> theta;
  real<lower=0> lambda;
}
transformed parameters{
  real eta;
  real zi;
  eta = log(lambda);
  zi = logit(theta);
}
model{
  theta ~ beta(1,1);
  lambda ~ normal(0,5);
  target += n0*log_sum_exp(bernoulli_logit_lpmf(1|zi),
bernoulli_logit_lpmf(0|zi)+poisson_log_lpmf(0|eta));
  target += np*bernoulli_logit_lpmf(0|zi) +
            poisson_log_lpmf(yp | eta);
}
"

zinb_bycatch <-"
data{
  int<lower=1> n0;
  int<lower=1> np;
  int<lower=1> yp[np];
}
parameters{
  real<lower=0,upper=1> theta;
  real<lower=0> mu;
  real<lower=0> r;
}
transformed parameters{
  real eta;
  eta=log(mu);
}
model{
  theta ~ beta(1,1);
  mu ~ normal(0,5);
  r ~ normal(0,5);
  target += n0*log_sum_exp(log(theta),
        log1m(theta)+neg_binomial_2_log_lpmf(0|eta,r));
  target += np*log1m(theta) +
            neg_binomial_2_log_lpmf(yp | eta, r);
}
"

##############################################
fit_zip <- stan_model(model_code = zip_bycatch)
fit_zinb <- stan_model(model_code = zinb_bycatch)

##save(fit_zinb, fit_zip, file="zip_nbstan.RData")
##load("zip_nbstan.RData")
##############################################
```

Processing data and initial values

```
############################################## The data and initial values
ZI_bycatch <- function(yp = zippos, n0 = zip0, model = "zip", n.chains = nchains) {
    np <- length(yp)
    N <- np + n0
    data <- list(np = np, n0 = n0, yp = yp)
    inits <- list()
    for (i in 1:n.chains) {
        if (model == "zip")
            inits[[i]] <- list(lambda = runif(1), theta = runif(1)) else inits[[i]] <- list(mu = runif(
    }
    if (model == "zip")
        paras <- c("lambda", "theta") else paras <- c("theta", "mu", "r")
    return(list(data = data, init = inits, nchains = n.chains, para = paras, model = model))
}

input.to.stan <- ZI_bycatch()
keep_zip <- sampling(fit_zip, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchains)  ##,
## control=list(max_treedepth=20))
print(keep_zip)
```

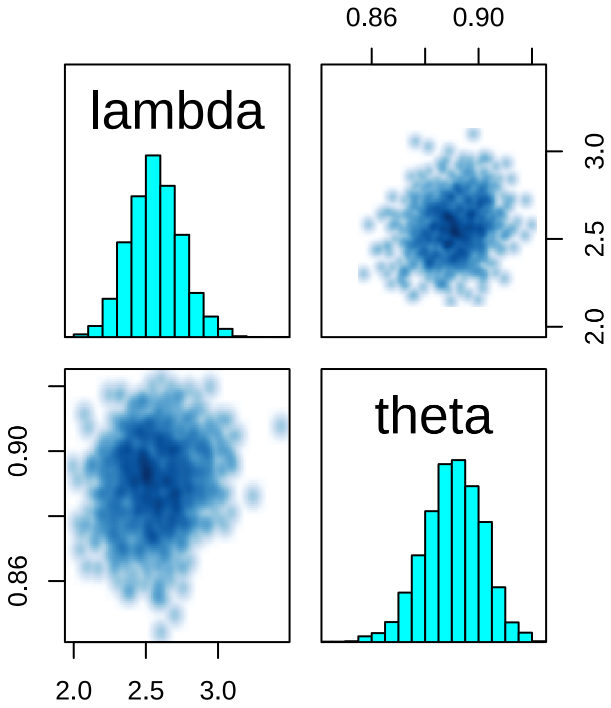
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## lambda    2.56    0.00 0.18    2.22    2.44    2.56    2.68    2.93  2555    1
## theta     0.89    0.00 0.01    0.87    0.88    0.89    0.90    0.91  2447    1
## lp__   -501.64    0.02 1.00 -504.35 -502.02 -501.33 -500.94 -500.68  2525    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:06:10 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
input.to.stan <- ZI_bycatch(model = "zinb")
keep_zinb <- sampling(fit_zinb, data = input.to.stan$data, init = input.to.stan$init,
    pars = input.to.stan$para, iter = niters, thin = nthin, chains = input.to.stan$nchains)  ##,
## control=list(max_treedepth=20))
print(keep_zinb)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## theta    0.62    0.00 0.19    0.11    0.52    0.68    0.76    0.84  1788    1
## mu       0.93    0.01 0.42    0.31    0.59    0.88    1.22    1.81  2008    1
## r        0.30    0.00 0.21    0.07    0.14    0.25    0.40    0.83  2112    1
## lp__  -456.91    0.03 1.33 -460.45 -457.53 -456.57 -455.93 -455.34  2154    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 14:06:40 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

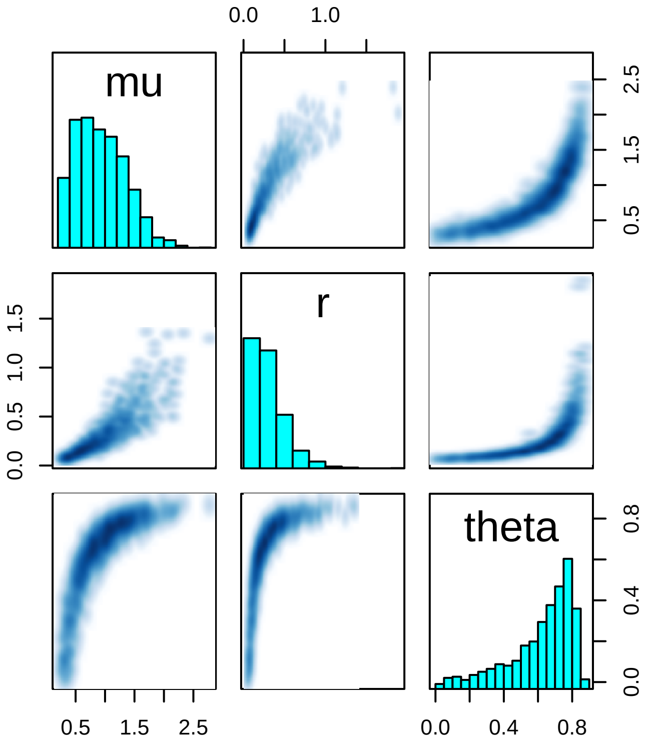
```
## save(keep_zip, keep_zinb,file='zinb_bycatch.RData')
pairs(keep_zip, pars = c("lambda", "theta"))
```

```
## 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, pars = c("mu", "r", "theta"))
```

```
## 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
```



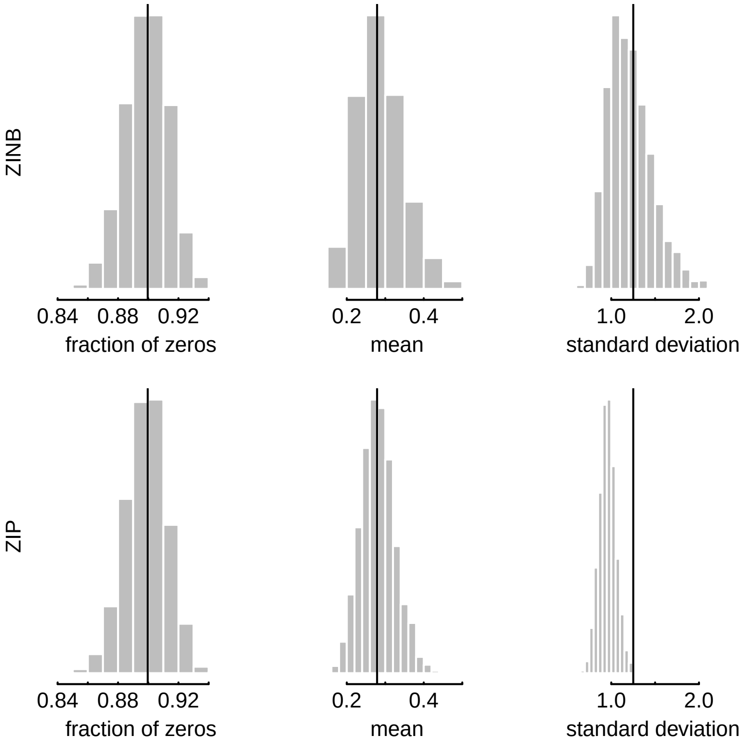
Strong correlation in the `zinb` model indicating non-identifiability. Additional information (e.g., of θ) is needed.

Making comparisons

```
zinb_coef <- as.data.frame(extract(keep_zinb, permute=T,
                                   pars=c("theta","mu","r")))
zip_coef <- as.data.frame(extract(keep_zip, permute=T,
                                  pars=c("theta","lambda")))

mu <- mean(c(rep(0,zip0),zippos))
s <- sd(c(rep(0,zip0),zippos))

## percent of 0 from zinb model
n0_zinb <- NULL
mu_zinb <- NULL
sig_zinb <- NULL
for (i in 1:dim(zinb_coef)[1]){
    n0tmp <- rbinom(1, n, zinb_coef$theta[i])
    tmp <- c(rep(0, n0tmp), rnbinom(n-n0tmp, mu=zinb_coef$mu[i],
                                    size=zinb_coef$r[i]))
    n0_zinb <- c(n0_zinb, mean(tmp==0))
    mu_zinb <- c(mu_zinb, mean(tmp))
    sig_zinb <- c(sig_zinb, sd(tmp))
}

## percent of 0 from zip model
```

```
n0_zip <- NULL
mu_zip <- NULL
sig_zip <- NULL
for (i in 1:dim(zip_coef)[1]){
    n0tmp <- rbinom(1, n, zip_coef$theta[i])
    tmp <- c(rep(0, n0tmp), rpois(n-n0tmp,
                                  lambda=zip_coef$lambda[i]))
    n0_zip <- c(n0_zip, mean(tmp==0))
    mu_zip <- c(mu_zip, mean(tmp))
    sig_zip <- c(sig_zip, sd(tmp))
}
```

```
par(mfrow=c(2,3), mar=c(2.5,2.5,1,1), mgp=c(1.25,0.125,0), tck=0.01)
hist(n0_zinb, xlim=range(c(n0_zinb, zip0/n, n0_zip)), yaxt="n",
     xlab="fraction of zeros", ylab="ZINB", main="",
     border="white", col="gray", density=-1)
abline(v=zip0/n)
## mean
hist(mu_zinb, xlim=range(c(mu_zinb, mu, mu_zip)), yaxt="n",
     xlab="mean", ylab="",  main="",
     border="white", col="gray", density=-1)
abline(v=mu)
## sd
hist(sig_zinb, xlim=range(c(sig_zinb, s, sig_zip)), yaxt="n",
     xlab="standard deviation", ylab="",  main="",
     border="white", col="gray", density=-1)
abline(v=s)

hist(n0_zip, xlim=range(c(n0_zip, zip0/n, n0_zinb)), yaxt="n",
     xlab="fraction of zeros", ylab="ZIP",  main="",
     border="white", col="gray", density=-1)
abline(v=zip0/n)
## mean
hist(mu_zip, xlim=range(c(mu_zip, mu, mu_zinb)), yaxt="n",
     xlab="mean", ylab="",  main="",
     border="white", col="gray", density=-1)
abline(v=mu)
## sd
hist(sig_zip, xlim=range(c(sig_zip, s, sig_zinb)), yaxt="n",
     xlab="standard deviation", ylab="",  main="",
     border="white", col="gray", density=-1)
abline(v=s)
```



Summary

In this example, we fit two common distributions (Poisson and Negative binomial) to model count data. In the first portion, we see that the Poisson model is not able to replicate important characteristics of the data while the negative binomial performs well. Increasing model complexity by accounting for zero-inflation improves the fit of both models, however, the ZINB model is not identifiable without additional information about theta. This exposes a bit of a model complexity trap, and highlights the importance of model evaluation to identify the most parsimonious approach.