Salamander example comparing Poisson, Zero-inflated Poisson, and Hurdle models

Mollie Brooks

2016-11-27

In this appendix, we reanalyze counts of salamanders in streams. Repeated samples of salamanders were taken at 23 sites. Some of the sites were affected by mountian top removal coal mining. The data was originally published in Price et al. (2016).

Preliminaries

Load packages

```
library(glmmTMB)
library(ggplot2); theme_set(theme_bw())
library(knitr)
library(bbmle) #for AICtab
library(reshape)
library(plyr)
```

Data organization

```
dat0 = read.csv("Price2015.csv")
dat1 = reshape(dat0, varying=list(3:6, 7:10,11:14,16:19, 20:23,24:27, 28:31, 32:35, 36:39, 40:43),
              timevar="sample", direction="long", sep=".")[,-15]
names(dat1) = c("site","mined", "cover", "sample", "DOP", "Wtemp", "DOY",
               "GP", "PR", "DM", "EC-A", "EC-L", "DES-L", "DF")
dat2 = melt(dat1, id=1:7)
names(dat2)[8:9] = c("spp", "count")
dat = transform(dat2,
               mined=factor(mined, levels=c(1,0), labels=c("yes", "no")),
               present=as.numeric(count>0)
)
head(dat)
```

```
##    site mined cover sample  DOP  Wtemp   DOY spp count present
## 1 VF -1   yes -1.44      1 -0.6 -1.229 -1.50  GP     0       0
## 2 VF- 2   yes  0.30      1 -0.6  0.085 -1.50  GP     0       0
## 3 VF -3   yes  0.40      1 -1.2  1.014 -1.29  GP     0       0
## 4  R -1    no -0.45      1  0.0 -3.023 -2.71  GP     2       1
## 5  R -2    no  0.60      1  0.6 -0.144 -0.69  GP     2       1
## 6  R -3    no  1.34      1  0.6 -0.015 -0.69  GP     1       1
```

Model fitting with glmmTMB

Poisson Models

The syntax for fitting Poisson models with `glmmTMB` is quite similar to using `glmer`. In the first model, the formula, `count~spp + (1|site)`, says that counts depend on species and vary randomly by site. We also pass it the data frame, `dat`, and specify a Poisson distribution using the `family` argument. `glmmTMB` assumes that we want a log-link with the Poisson distribution because that's the standard.

```
pm0 = glmmTMB(count~spp + (1|site), dat, family="poisson")
pm1 = glmmTMB(count~spp + mined + (1|site), dat, family="poisson")
pm2 = glmmTMB(count~spp * mined + (1|site), dat, family="poisson")
```

Zero-inflated Poisson

To fit zero-inflated models, we use the `ziformula` argument, or `glmmTMB` will also recognize `zi` This is a formula that describes how the probability of an extra zero (i.e. structural zero) will vary with any predictors. In this example, we might assume that absences will at least vary by species (`spp`), so we write `zi=~spp`. This formula only has a right side because the left side is always the probability of having a structural zero in the response that was specified in the first formula. The zero-inflation probability is always modeled with a logit-link to keep it between 0 and 1.

```
zipm0 = glmmTMB(count~spp + (1|site), zi=~spp, dat, family="poisson")
zipm1 = glmmTMB(count~spp + mined + (1|site), zi=~spp, dat, family="poisson")
zipm2 = glmmTMB(count~spp + mined + (1|site), zi=~spp + mined, dat, family="poisson")
zipm3 = glmmTMB(count~spp * mined + (1|site), zi=~spp * mined, dat, family="poisson")
```

Poisson hurdle model

We can also fit hurdle models. This is done in two parts: first by modelling the zeros versus non-zeros with a binomial distribution and then modelling the non-zeros with a truncated-Poisson distribution. In the salamander example, this means first modeling presence versus absence (`zm1` below) and then modeling the presence only data (`cm1` below). In `zm1`, it's important that the response `present` is numeric (`1` or `0`) instead of (`TRUE` or `FALSE`); that's why we wrote `present=as.numeric(count>0)` in the data organization code above.

```
zm0 = glmmTMB(present~spp, family="binomial", dat)
zm1 = glmmTMB(present~spp + mined, family="binomial", dat)
zm2 = glmmTMB(present~spp * mined, family="binomial", dat)
AICtab(zm0, zm1, zm2)
```

```
##     dAIC  df
## zm2   0.0 14
## zm1   4.3 8
## zm0 167.2 7
```

For the second part of the hurdle model, we look at the presence data, `subset(dat, present==1)`.

```
cm0 = glmmTMB(count~spp + (1|site),
  family=list(family="truncated_poisson", link="log"),
  data=subset(dat, present==1))
cm1 = glmmTMB(count~spp + mined + (1|site),
  family=list(family="truncated_poisson", link="log"),
  data=subset(dat, present==1))
```

```r
cm2 = glmmTMB(count~spp * mined + (1|site),
  family=list(family="truncated_poisson", link="log"),
  data=subset(dat, present==1))
AICtab(cm0, cm1, cm2)
```

```
##     dAIC df
## cm2  0.0 15
## cm1  2.8 9
## cm0 26.2 8
```

Model comparison using AIC

We can use `AICtab` to compare the Poisson and zero-inflted Poisson models. To get the AIC value of hurdle models, we sum the AIC values of the two parts of the model. Then we can compare this AIC value to that of our best Poisson model.

```r
AICtab(pm0, pm1, pm2, zipm0, zipm1, zipm2, zipm3)
```

```
##       dAIC  df
## zipm3   0.0 29
## zipm2   7.4 17
## zipm1  24.8 16
## zipm0  54.4 15
## pm2   162.2 15
## pm1   184.7 9
## pm0   214.6 8
```

```r
AIC(zm2)+AIC(cm2)
```

```
## [1] 1783
```

```r
AIC(zipm3)
```

```
## [1] 1778
```

The zero-inflated Poisson model has a slightly lower AIC value than the hurdle model.

Model summary

```r
summary(zipm3)
```

```
##  Family: poisson  ( log )
## Formula: count ~ spp * mined + (1 | site)
##    Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##     1778     1908     -860     1720      615
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  site   (Intercept) 0.103    0.321
```

```
## Number of obs: 644, groups:  site, 23
##
## Conditional model:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.104      1.012   -0.10     0.92
## sppPR              -0.222      1.191   -0.19     0.85
## sppDM               0.500      1.015    0.49     0.62
## sppEC-A            -0.689      1.368   -0.50     0.61
## sppEC-L            -1.120      1.211   -0.92     0.36
## sppDES-L            0.432      1.005    0.43     0.67
## sppDF               0.260      1.013    0.26     0.80
## minedno             1.040      1.021    1.02     0.31
## sppPR:minedno      -0.425      1.229   -0.35     0.73
## sppDM:minedno      -0.292      1.024   -0.28     0.78
## sppEC-A:minedno     0.498      1.383    0.36     0.72
## sppEC-L:minedno     1.822      1.218    1.50     0.13
## sppDES-L:minedno    0.168      1.014    0.17     0.87
## sppDF:minedno      -0.261      1.026   -0.26     0.80
##
## Zero-inflation model:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)          3.22       1.19    2.70  0.00696 **
## sppPR               -1.34       1.44   -0.93  0.35118
## sppDM               -2.22       1.26   -1.76  0.07829 .
## sppEC-A             -1.73       1.63   -1.06  0.29038
## sppEC-L             -4.54       3.39   -1.34  0.18057
## sppDES-L            -2.59       1.26   -2.05  0.04028 *
## sppDF               -2.97       1.28   -2.33  0.01982 *
## minedno             -4.90       1.30   -3.78  0.00016 ***
## sppPR:minedno        3.79       1.59    2.39  0.01698 *
## sppDM:minedno        2.60       1.41    1.84  0.06530 .
## sppEC-A:minedno      3.67       1.74    2.11  0.03505 *
## sppEC-L:minedno      5.10       3.45    1.48  0.13959
## sppDES-L:minedno     2.83       1.41    2.00  0.04527 *
## sppDF:minedno        3.67       1.43    2.58  0.00998 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Plotting model results

Now we can plot estimates from the best model. It's easiest to see these using the `predict` function. To avoid marginalizing over or conditioning on random effects, we refit the best model without the random effect of site.

```
zipm3FE = glmmTMB(count~spp * mined, zi=~spp * mined, dat, family="poisson")
newdata0 = newdata = unique(dat[,c("mined","spp")])
temp = predict(zipm3FE, newdata, se.fit=TRUE, zitype="response")
newdata$predFE = temp$fit
newdata$predFE.min = temp$fit-1.98*temp$se.fit
newdata$predFE.max = temp$fit+1.98*temp$se.fit

real=ddply(dat, ~site+spp+mined, summarize, m=mean(count))
```

```
ggplot(newdata, aes(spp, predFE, colour=mined))+geom_point()+
  geom_errorbar(aes(ymin=predFE.min, ymax=predFE.max))+
  geom_point(data=real, aes(x=spp, y=m) )+
  ylab("Average abundance \n including presences and absences")
```

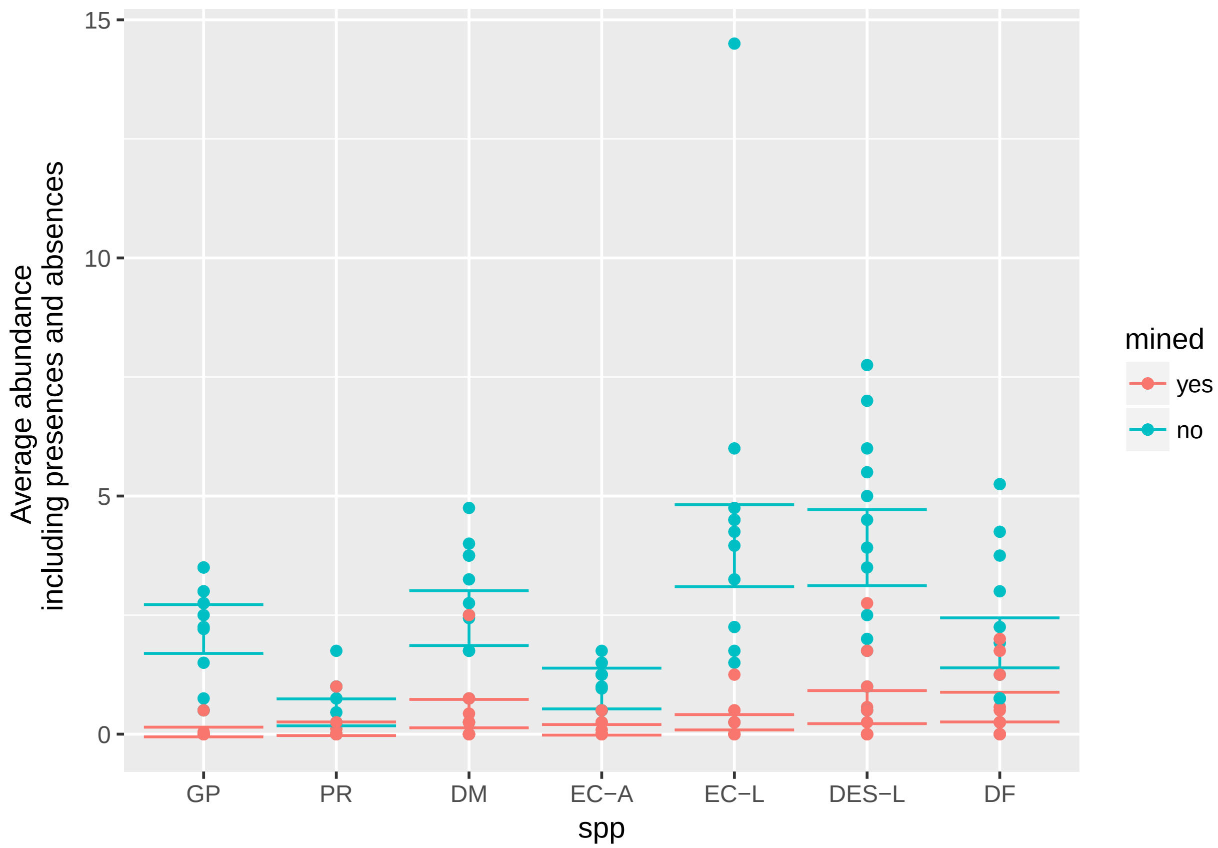


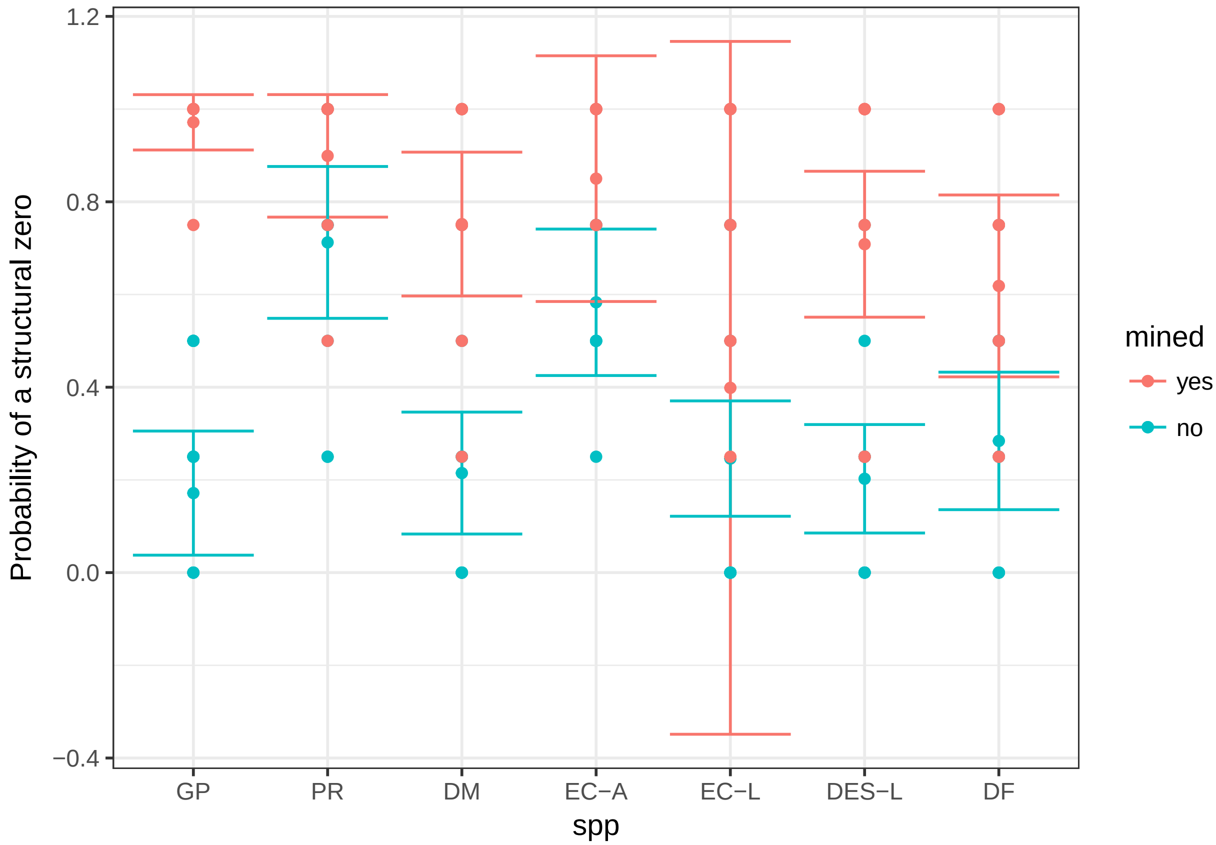
Figure 1 - Points represtent site-specific average counts. Error bars represent the 95% Wald-type confidence intervals for the predicted average count.

We can also get predicted probability of a structural zero using the argument `zitype="zprob"`.

```
temp2 = predict(zipm3FE, newdata, se.fit=TRUE, zitype="zprob")
newdata$predZ = temp2$fit
newdata$predZ.min = temp2$fit-1.98*temp2$se.fit
newdata$predZ.max = temp2$fit+1.98*temp2$se.fit

real = ddply(dat, ~site+spp+mined, summarize, absence=mean(count==0))

ggplot(newdata, aes(spp, predZ, colour=mined))+geom_point()+
  geom_errorbar(aes(ymin=predZ.min, ymax=predZ.max))+
  geom_point(data=real, aes(x=spp, y=absence) )+
  ylab("Probability of a structural zero")
```



Wald-type confidence intervals are not really appropriate for probabiliities because they don't take the boundaries (0, 1) into account.

alternative prediction methods

As previously discussed in various places there are a whole bunch of decisions to make Now we want the predicted value and confidence intervals on `(1-prob)*count`.

see here

Setting random effects to zero (i.e., predicting at population mode):

```
X.cond <- model.matrix(lme4::nobars(formula(zipm3)[-2]), newdata0)
beta.cond <- fixef(zipm3)$cond
pred.cond <- X.cond %*% beta.cond
predvar.cond <- diag(X.cond %*% vcov(zipm3)$cond %*% t(X.cond))
predse.cond <- sqrt(predvar.cond)
ziformula <- zipm3$modelInfo$allForm$ziformula
X.zi <- model.matrix(lme4::nobars(ziformula), newdata0)
beta.zi <- fixef(zipm3)$zi
pred.zi <- X.zi %*% beta.zi
predvar.zi <- diag(X.zi %*% vcov(zipm3)$zi %*% t(X.zi))
predse.zi <- sqrt(predvar.zi)
```

these are estimates of the linear predictors (i.e., predictions on the link scale: logit(prob) and log(cond)), not the predictions themselves. The easiest thing to do for the point estimates of the unconditional cond (`ucount`) is just to transform to the response scale and multiply:

```
pred.ucount <- exp(pred.cond)*(1-plogis(pred.zi))
```

For the standard errors/confidence intervals, we could use posterior predictive simulations (i.e. draw MVN samples from the parameter for the fixed effects) (this conditions on/ignores uncertainty in the random-effect parameters …)

```
library(MASS)
set.seed(101)
pred.condpar.psim <- mvrnorm(1000,mu=beta.cond,Sigma=vcov(zipm3)$cond)
pred.zipar.psim <- mvrnorm(1000,mu=beta.zi,Sigma=vcov(zipm3)$zi)
pred.cond.psim <- X.cond %*% t(pred.condpar.psim)
pred.zi.psim <- X.zi %*% t(pred.zipar.psim)
pred.ucount.psim <- exp(pred.cond.psim)*(1-plogis(pred.zi.psim))
ci.ucount <- t(apply(pred.ucount.psim,1,quantile,c(0.025,0.975)))
ci.ucount <- data.frame(ci.ucount)
names(ci.ucount) <- c("ucount.low","ucount.high")
print(ci.ucount,digits=2)
```

```
##     ucount.low ucount.high
## 1       0.0018        0.54
## 4       1.5107        2.96
## 93      0.0122        0.65
## 96      0.1797        0.86
## 185     0.1586        0.93
## 188     1.7098        3.29
## 277     0.0051        0.97
## 280     0.5385        1.45
## 369     0.0017        1.01
## 372     2.8398        5.14
## 461     0.1806        1.15
## 464     2.8129        4.83
## 553     0.1983        1.12
## 556     1.2084        2.55
```

We could also use the Delta method for this.

Simulating from a fitted model

We could also look at the distribution of simulated values from the best fitted model. For this we use the function `simulate.glmmTMB`.

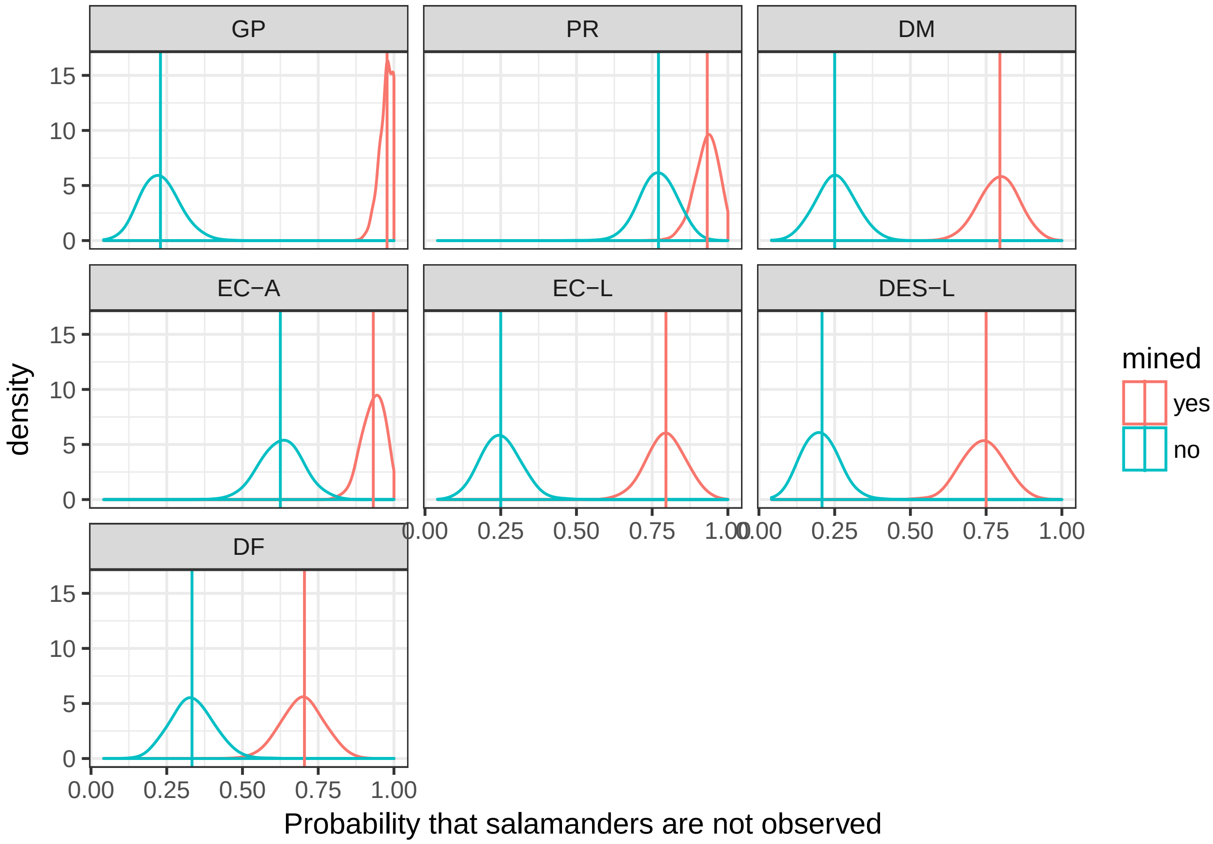
```
sims=simulate(zipm3, seed = 1, nsim = 1000)
```

This function returns a list of vectors. The list has one element for each simulation (`nsim`) and the vectors are the same shape as our response variable.

```
simdatlist=lapply(sims, function(count){
  cbind(count, dat[,c('site', 'mined', 'spp')])
})
simdatsums=lapply(simdatlist, function(x){
  ddply(x, ~spp+mined, summarize, absence=mean(count==0))
})
ssd=do.call(rbind, simdatsums)
```

Then we can plot them with the observations summarized in the same way.

```r
real = ddply(dat, ~spp+mined, summarize, absence=mean(count==0))
ggplot(ssd, aes(x=absence, color=mined))+
  geom_density(adjust=2)+
  facet_wrap(~spp)+
  geom_vline(data=real, aes(xintercept=absence, color=mined))+
  xlab("Probability that salamanders are not observed")
```



We can see that this model does a good job of capturing the observed zero counts.

Price, Steven J., Brenee' L. Muncy, Simon J. Bonner, Andrea N. Drayer, and Christopher D. Barton. 2016. "Effects of Mountaintop Removal Mining and Valley Filling on the Occupancy and Abundance of Stream Salamanders." *Journal of Applied Ecology* 53 (2): 459–68. doi:10.1111/1365-2664.12585.