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

Mollie Brooks

2016-12-09

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) and was aquired from Dryad (Price et al. (2015)).

Preliminaries

Load packages

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

Load and organize data

```
data(Salamanders)
head(Salamanders)
```

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

```
Salamanders = transform(Salamanders, present = as.numeric(count>0))
```

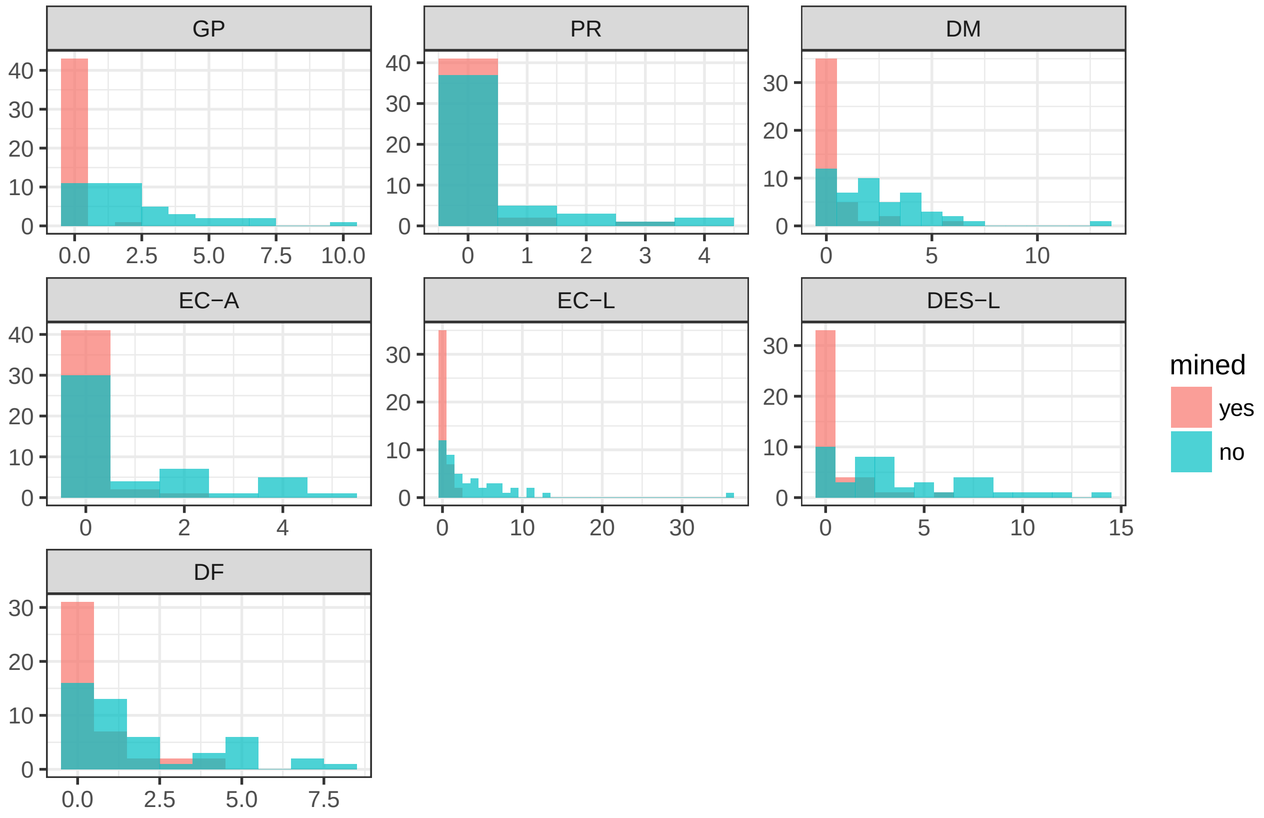

Figure 1 – Observed count data. Histograms of count data split into separate panels for each species or life stage. Each panel contains two overlaid histograms in which color represents whether the site was mined or not.

Model fitting with glmmTMB

These analyses are intended to be a simple demonstration of how to use some features of the `glmmTMB` package, so we do not attempt to fit all of the models that could be considered reasonable.

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, `Salamanders`, 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), Salamanders, family="poisson")
pm1 = glmmTMB(count~spp + mined + (1|site), Salamanders, family="poisson")
pm2 = glmmTMB(count~spp * mined + (1|site), Salamanders, 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`) and vary randomly by site, so we write `zi=~spp + (1|site)`. 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 +(1|site), Salamanders, family="poisson")
zipm1 = glmmTMB(count~spp + mined +(1|site), zi=~spp +(1|site), Salamanders, family="poisson")
zipm2 = glmmTMB(count~spp + mined +(1|site), zi=~spp + mined +(1|site), Salamanders, family="poisson")
zipm3 = glmmTMB(count~spp * mined +(1|site), zi=~spp * mined +(1|site), Salamanders, 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 + (1|site), family="binomial", Salamanders)
zm1 = glmmTMB(present~spp + mined + (1|site), family="binomial", Salamanders)
zm2 = glmmTMB(present~spp * mined + (1|site), family="binomial", Salamanders)
AICtab(zm0, zm1, zm2)
```

```
##     dAIC df
## zm2  0.0 15
## zm1  5.9 9
## zm0 31.6 8
```

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

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

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

```
##       dAIC df
## zipm3   0  30
## zipm2  14  18
## zipm1  32  17
## zipm0  58  16
## pm2   188  15
## pm1   211  9
## pm0   240  8
```

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

```
## [1] 1759
```

```
AIC(zipm3)
```

```
## [1] 1752
```

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

Model summary

```
summary(zipm3)
```

```
##  Family: poisson  ( log )
## Formula:          count ~ spp * mined + (1 | site)
## Zero inflation:         ~ ~spp * mined + (1 | site)
## Data: Salamanders
##
##      AIC      BIC   logLik deviance df.resid
##     1752     1886     -846     1692      614
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  site   (Intercept) 0.0814   0.285
## Number of obs: 644, groups:  site, 23
##
## Zero-inflation model:
##  Groups Name        Variance Std.Dev.
##  site   (Intercept) 1.06     1.03
## Number of obs: 644, groups:  site, 23
##
## Conditional model:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.185      0.948    0.20    0.845
## sppPR               -0.222      1.141   -0.19    0.846
## sppDM                0.340      0.972    0.35    0.726
## sppEC-A             -0.804      1.338   -0.60    0.548
## sppEC-L             -1.618      0.987   -1.64    0.101
## sppDES-L             0.377      0.960    0.39    0.694
## sppDF                0.108      0.968    0.11    0.911
## minedno              0.746      0.957    0.78    0.436
## sppPR:minedno       -0.365      1.178   -0.31    0.757
## sppDM:minedno       -0.117      0.982   -0.12    0.905
## sppEC-A:minedno      0.590      1.354    0.44    0.663
## sppEC-L:minedno     2.338      0.996    2.35    0.019 *
## sppDES-L:minedno    0.228      0.970    0.24    0.814
## sppDF:minedno      -0.156      0.984   -0.16    0.874
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Zero-inflation model:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)          4.06       1.23    3.29   0.0010 **
## sppPR               -1.42       1.43   -1.00   0.3186
## sppDM               -2.67       1.27   -2.10   0.0357 *
## sppEC-A             -1.97       1.72   -1.14   0.2539
## sppEC-L            -21.57    2785.59   -0.01   0.9938
## sppDES-L            -2.97       1.26   -2.36   0.0183 *
## sppDF               -3.55       1.31   -2.72   0.0065 **
## minedno             -6.05       1.41   -4.30 0.000017 ***
## sppPR:minedno        4.39       1.61    2.72   0.0065 **
## sppDM:minedno        3.18       1.45    2.19   0.0284 *
## sppEC-A:minedno      4.21       1.87    2.26   0.0241 *
## sppEC-L:minedno     22.28    2785.59    0.01   0.9936
## sppDES-L:minedno     3.30       1.44    2.29   0.0222 *
## sppDF:minedno        4.18       1.53    2.74   0.0062 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This summary can be broken down into four sections. The top section is a general overview containing a description of the model specification (`Family`, `Formula`, `Zero inflation`, `Dispersion`, `Data`) and resulting information criteria. The information criteria are only meaningful in comparison to other models fit by `glmmTMB`; this is because `glmmTMB` does not drop any constants from the likelihood while some other packages do. The second section describes the variability of the `Random effects`. In this model, we had random effects on both the conditional model and the zero-inflation model. The third section describes the coefficients of the `Conditional model` including Wald p-values. Besides the intercept, the estimates are all contrasts as is standard in regression models. This model has a log link as stated in the top line of the summary. The fourth section describes the `Zero-inflation model` similarly to the `Conditional model` except that this model has a logit-link. The zero-inflation model estimates the probability of an extra zero such that, a positive contrast indicates a higher chance of absence (e.g. `minedno` < 0 means fewer absences in sites unafected by mining); this is the opposite of the conditional model where a positive contrast indicates a higher abundance (e.g. `minedno` > 0 means higher abundances in sites unafected by mining).

Plotting model results

As previously discussed in various places there are a whole bunch of decisions to make about marginalizing over or conditioning on the random effects. See discussion at this link.

Quick and dirty plot

It's easiest to see the pattern by using the `predict` function. To avoid marginalizing over or conditioning on random effects, we can refit the best model without the random effect of site; however, this is not ideal because it ignores the correlation within sites. We present a more rigorous version next.

```
zipm3FE = glmmTMB(count~spp * mined, zi=~spp * mined, Salamanders, family="poisson")
newdata0 = newdata = unique(Salamanders[,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(Salamanders, ~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")+
  xlab("Species")
```

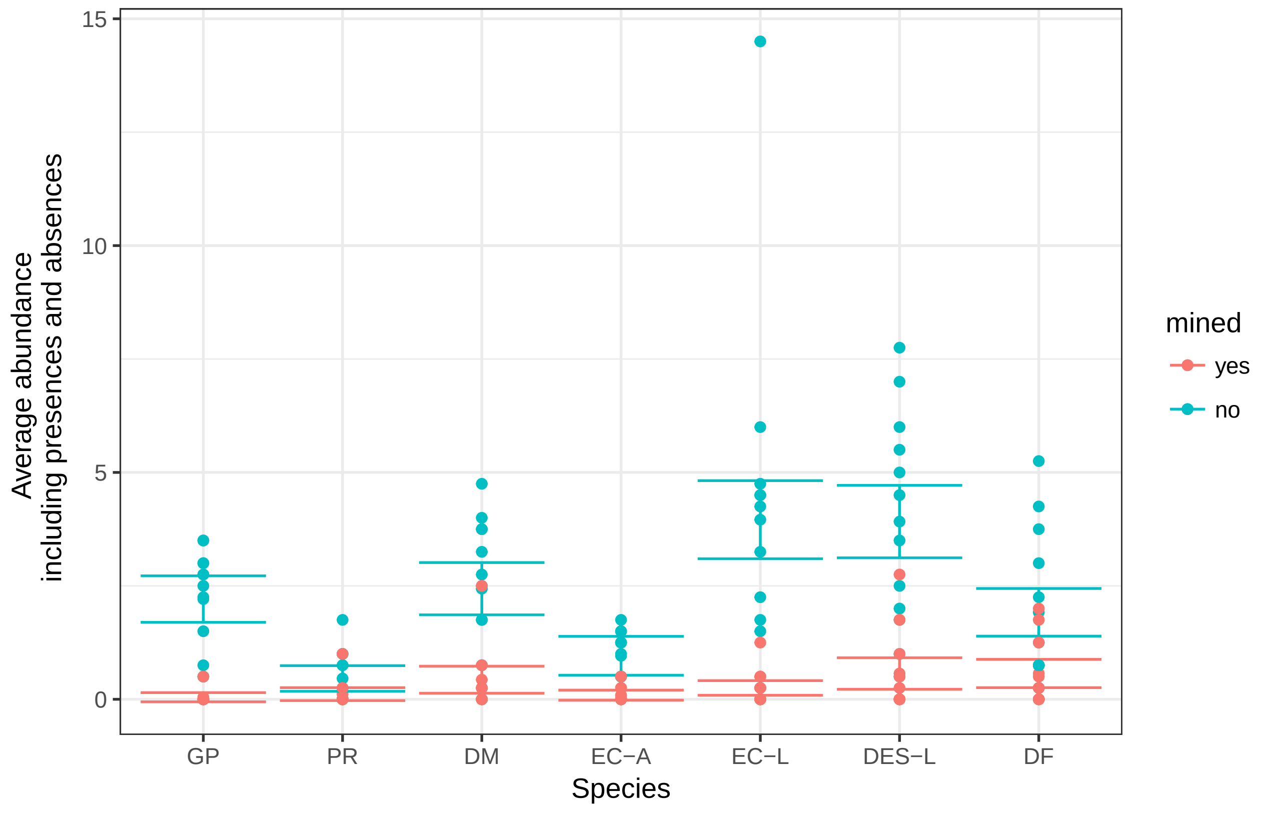

Figure 2 – Estimated abundance ignoring correlation. Points represtent site-specific average counts. Error bars represent the 95% Wald-type confidence intervals for the predicted average count.

Alternative prediction method

We can predict at the population mode, by setting the random effects to zero.

```
X.cond = model.matrix(lme4::nobars(formula(zipm3)[-2]), newdata0)
beta.cond = fixef(zipm3)$cond
pred.cond = X.cond %*% beta.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
```

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 count (`ucount`) is to transform to the response scale and multiply:

```
pred.ucount = exp(pred.cond)*(1-plogis(pred.zi))
pred.count = data.frame(newdata0, pred.ucount)
```

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")
pred.count = data.frame(newdata0, pred.ucount, ci.ucount)
```

These predicted counts should be close to the median counts, so we plot them together to compare.

```
real.count = ddply(Salamanders, ~spp+mined, summarize, m=median(count), mu=mean(count))
ggplot(pred.count, aes(x=spp, y=pred.ucount, colour=mined))+geom_point(shape=1, size=2)+
  geom_errorbar(aes(ymin=ucount.low, ymax=ucount.high))+
  geom_point(data=real.count,  aes(x=spp, y=m, colour=mined), shape=0, size=2)+
  geom_point(data=real.count,  aes(x=spp, y=mu, colour=mined), shape=5, size=2)+
  ylab("Abundance \n including presences and absences")+
  xlab("Species")
```

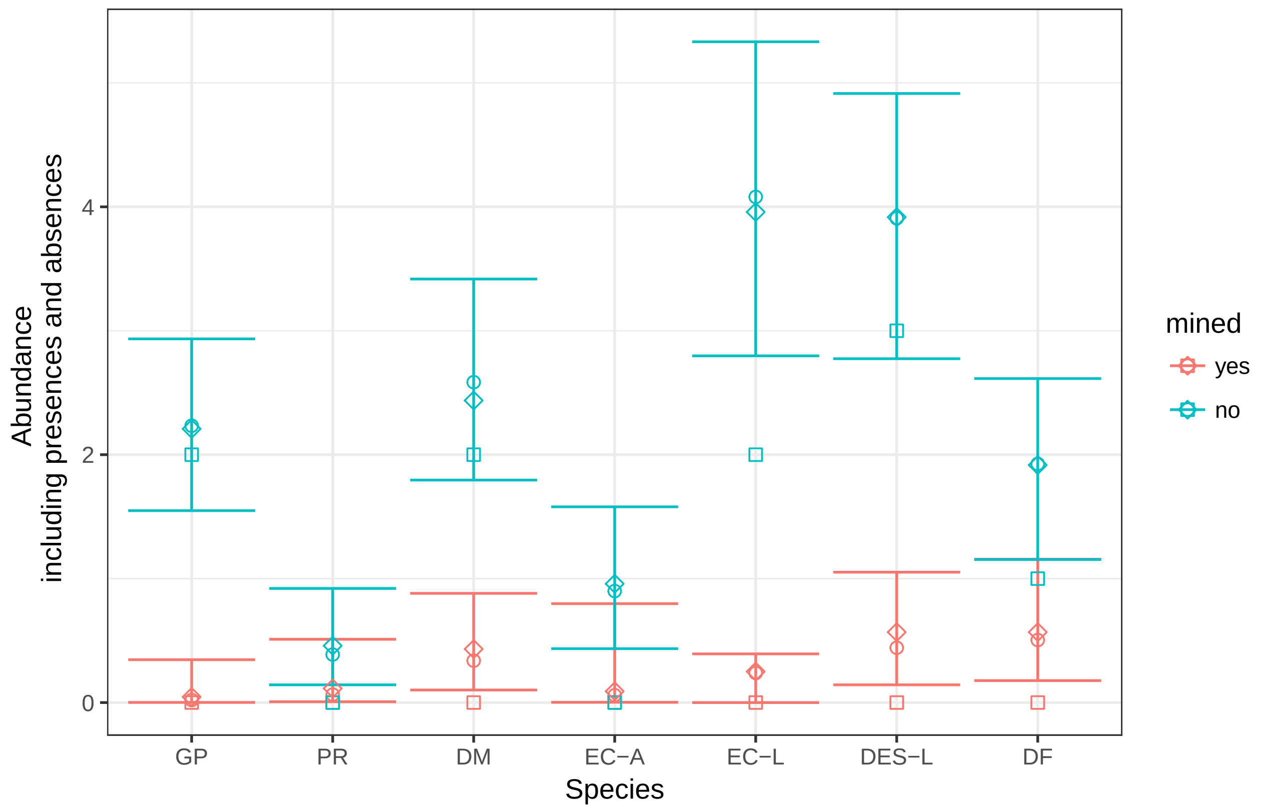

Figure 3 – Estimated abundance at mode. Circles represent predicted unconditional counts at the mode (i.e., site effect = 0) and error bars represent the 95% confidence intervals for that mode. Squares represnet the observed median and diamonds represent observed means calculated across samples and sites. In this highly skewed data, the mode is closer to the mean than the median.

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

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

```
real = ddply(Salamanders, ~spp+mined, summarize,
             absence=mean(count==0),
             mu=mean(count))
ggplot(ssd, aes(x=absence, color=mined))+
  geom_density(adjust=4)+
  facet_wrap(~spp)+
  geom_point(data=real, aes(x=absence, y=1, color=mined), size=2)+
  xlab("Probability that salamanders are not observed")+ylab(NULL)
```

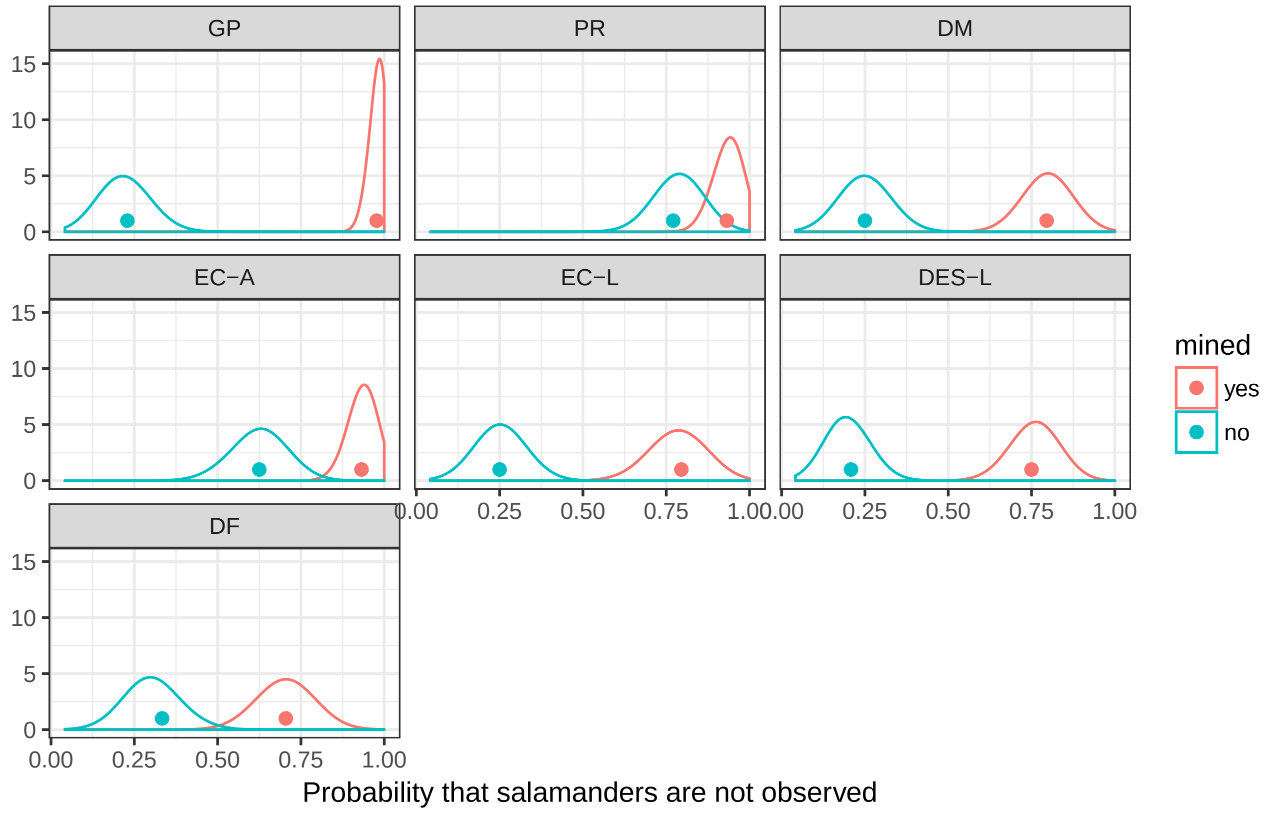

Figure 4 – Simulated zero counts. Each panel represents a different species or life stage of a species. Densities are values from 1000 data sets simulated from our best fit model. Points represent the observed data.

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

```
ggplot(ssd, aes(x=mu, color=mined))+
  geom_density(adjust=4)+
  facet_wrap(~spp)+
  geom_point(data=real, aes(x=mu, y=.5, color=mined), size=2)+
  xlab("Abundance including presences and absences")+ylab(NULL)
```

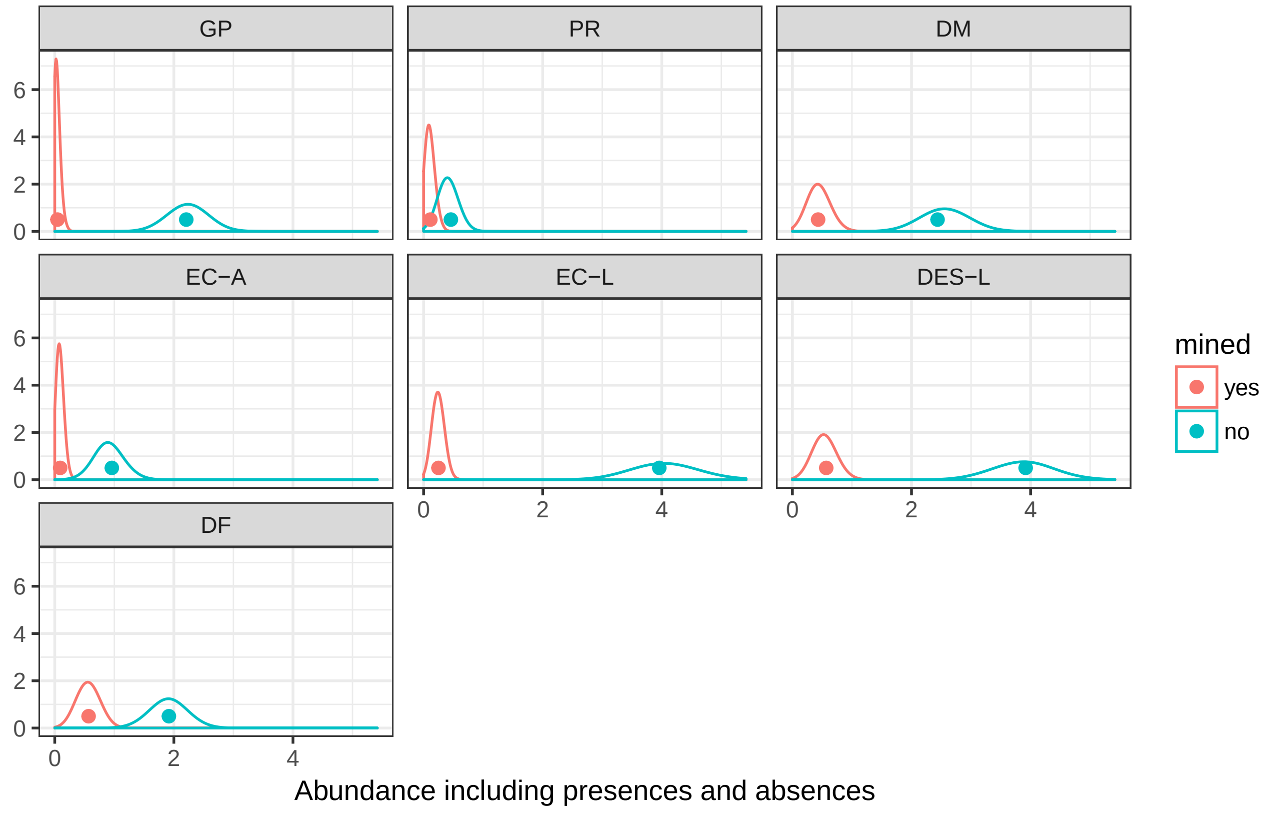

Figure 5 – Simulated unconditional abundances. Each panel represents a different species or life stage of a species. Densities are values from 1000 data sets simulated from our best fit model. Points represent the observed data.

References

Price, S J, B L Muncy, S J Bonner, A N Drayer, and C D Barton. 2015. "Data from: Effects of Mountaintop Removal Mining and Valley Filling on the Occupancy and Abundance of Stream Salamanders." *Journal of Applied Ecology*. Dryad Digital Repository. doi:doi:10.5061/dryad.5m8f6.

———. 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.