# Experimental Support for Random Effects in unmarked

Ken Kellner

December 1, 2021

## 1 Introduction

Random effects are often useful when fitting hierarchical ecological models. For example, if we have many different observers collecting count or presence data, it may be appropriate to include observer as a random effect on the detection process. If we have multiple years of data at our sites, and are fitting a so-called "stacked" model where site-years are the response data instead of sites, it may be appropriate to include site as a random effect on the state process. Until recently, including random effects as part of the linear predictor for the state or detection parameter in a model was possible only by fitting the model in a Bayesian framework with e.g. WinBUGS or JAGS.

The `unmarked` package now includes experimental support for fitting models with random effects, via the use of Template Model Builder (TMB). TMB uses Laplace approximation to estimate the random effects. Currently, only a few model types including single-season occupancy (`occu`) and $N$-mixture models (`pcount`) have random effects support, but more models may be supported in the future.

### 1.1 Caveats

The first caveat is that `unmarked` can only fit normally-distributed random effects with mean 0 on the link scale, and nested and correlated random effects are not supported. Second, we have found that estimation of random effects in `unmarked` via TMB works well in many cases, but for datasets with small sample sizes or sparse (many 0) observations, estimates are often poor and can result in misleading inference. We urge `unmarked` users to incorporate random effects with caution, and compare results to similar models without random effects. Some datasets may simply not be appropriate for models with random effects in `unmarked`. In these cases Bayesian methods may be more appropriate.

## 2 Example model with random effects

Below, we demonstrate fitting an $N$-mixture model via `pcount` to a dataset, including different combinations of random effects.

### 2.1 Simulating a dataset

We begin by simulating a count dataset, in which 100 sites have been visited 3 times per year in each of 3 years. Abundance will be a function of a single fixed-effect covariate, and we will consider site a random effect. We will simulate detection with a random observer effect.

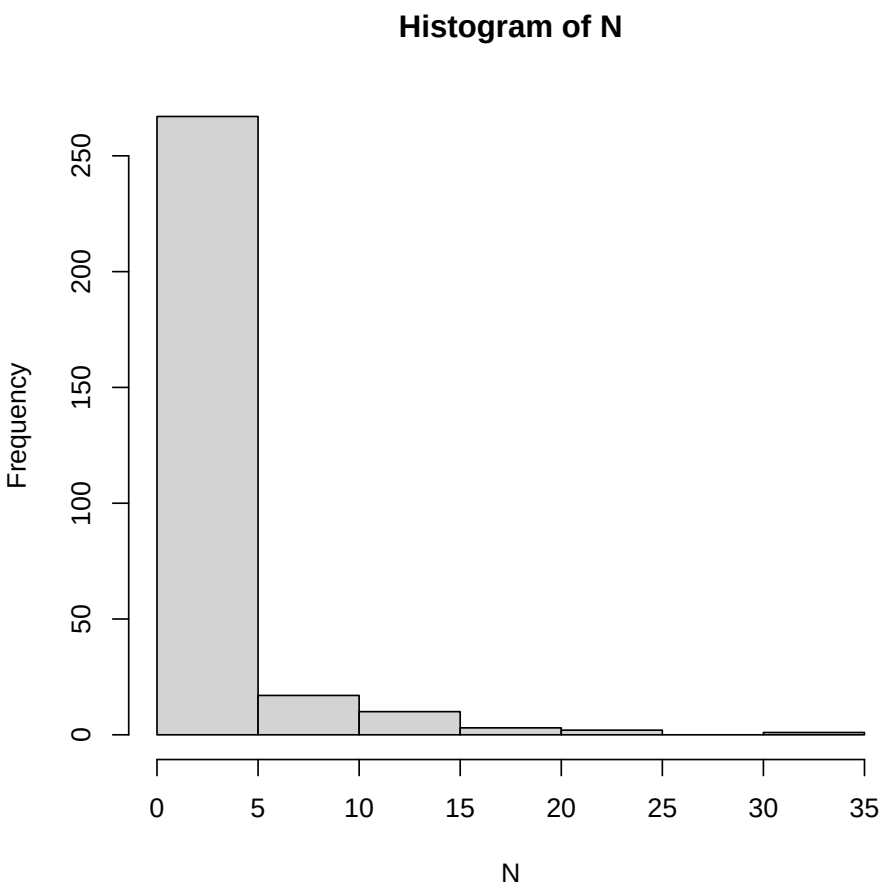
First, define the simulation parameters, covariate data, and the random effect values:

```
> set.seed(35)
> nsites <- 100
> nyears <- 3
> nvisits <- 3
> # Abundance parameters
> beta0 <- 0 # Intercept
> beta1 <- 1 # fixed covariate slope
> sd_site <- 0.5 # SD of site-level random effect
> re_site <- rnorm(nsites, 0, sd_site) # simulate random effect
> # Detection parameters
> alpha0 <- 0 # Intercept
```

```
> sd_obs <- 0.3 # SD of observer-level random effect (20 unique observers)
> re_obs <- rnorm(20, 0, sd_obs) # simulate random effect
> # Covariates
> x <- rnorm(nsites*nyears) # a covariate on abundance
> site_id <- rep(1:100, each=3)
> obs_id <- sample(1:20, nsites*nyears*nvisits, replace=TRUE)
```

Next, calculate $\lambda$ for each site and simulate the actual abundance `N`:

```
> lambda <- exp(beta0 + beta1*x + # fixed part of linear predictor
              re_site[site_id])  # site random effect
> # Generate latent abundance N
> N <- rpois(nsites*nyears, lambda)
> hist(N)
```




Simulate `p` by observer, and using `N`, simulate the observed counts `y`:

```
> p <- plogis(alpha0 + re_obs[obs_id])
> p <- matrix(p, nrow=nsites*nyears, ncol=nvisits, byrow=TRUE)
> y <- matrix(NA, nsites*nyears, nvisits)
> for (i in 1:(nsites*nyears)){
   for (j in 1:nvisits){
     y[i,j] <- rbinom(1, N[i], p[i,j])
   }
 }
```

Finally, organize the data into an `unmarkedFrame`. Note that we are specifying our random effect parameters as R factors.

```
> library(unmarked)
> site_covs <- data.frame(x=x,
                          site_id=factor(as.character(site_id)))
> obs_covs <- data.frame(obs_id=factor(as.character(obs_id)))
> umf <- unmarkedFramePCount(y=y, siteCovs=site_covs, obsCovs=obs_covs)
```

## 2.2 Fitting models

First we will fit a model without random effects, including only the fixed effect covariate `x`.

```
> mod_x <- pcount(~1~x, umf, K=40)
> summary(mod_x)
Call:
pcount(formula = ~1 ~ x, data = umf, K = 40)

Abundance (log-scale):
            Estimate     SE     z   P(>|z|)
(Intercept)    0.313 0.0656  4.76  1.91e-06
x              0.887 0.0381 23.26 1.13e-119

Detection (logit-scale):
 Estimate     SE   z P(>|z|)
    0.209 0.0953 2.2  0.0281

AIC: 2128.396
Number of sites: 300
optim convergence code: 0
optim iterations: 25
Bootstrap iterations: 0
```

This model overestimates the abundance intercept (truth = 0) and underestimates the effect of `x` (truth = 1). Next, we will fit a model with random intercepts by site. This is specified in the abundance formula, using syntax similar to that found in package `lme4`. To specify random intercepts based on our site covariate `site_id`, add `(1|site_id)` to the abundance formula (the parentheses are required). You can read this as "the intercepts should be random based on `site_id`". Note that the fixed effect `x` is outside the parentheses in the abundance formula.

```
> mod_site <- pcount(~1~x+(1|site_id), umf, K=40)
> mod_site
Call:
pcount(formula = ~1 ~ x + (1 | site_id), data = umf, K = 40)

Abundance:
Random effects:
  Groups        Name Variance Std.Dev.
 site_id (Intercept)     0.32    0.566

Fixed effects:
            Estimate    SE     z  P(>|z|)
(Intercept)    0.210 0.101  2.07 3.81e-02
x              0.977 0.056 17.42 5.37e-68

Detection:
 Estimate    SE      z P(>|z|)
   -0.109 0.121 -0.906   0.365

AIC: 1996.631
```

In the summary output `unmarked` provides an estimate of the random effect SD, which is similar to the true value (0.5). The other abundance parameters are also now closer to their true values. Now we'll fit the model with random detection intercepts by observer, in addition to the site random effect. As before, we add `(1|obs_id)` to the detection formula.

```
> mod_obs <- pcount(~1 + (1|obs_id) ~ x + (1|site_id), umf, K=40)
> mod_obs
Call:
pcount(formula = ~1 + (1 | obs_id) ~ x + (1 | site_id), data = umf,
    K = 40)

Abundance:
Random effects:
  Groups        Name Variance Std.Dev.
 site_id (Intercept)     0.31    0.557

Fixed effects:
            Estimate     SE     z  P(>|z|)
(Intercept)    0.186 0.0996  1.87 6.17e-02
x              0.967 0.0561 17.24 1.30e-66

Detection:
Random effects:
 Groups        Name Variance Std.Dev.
 obs_id (Intercept)    0.062    0.249

Fixed effects:
 Estimate    SE      z P(>|z|)
  -0.0368 0.134 -0.275   0.783

AIC: 1989.882
```

Both estimated random effect SDs are similar to the "true" values.

## 2.3 Model inference

To get more details, including 95% CIs, on the random effects SDs, use the `sigma` function:

```
> sigma(mod_obs)
  Model  Groups        Name     sigma     lower    upper
1   lam site_id (Intercept) 0.5572061 0.4429796 0.700887
2     p  obs_id (Intercept) 0.2492400 0.1393307 0.445850
```

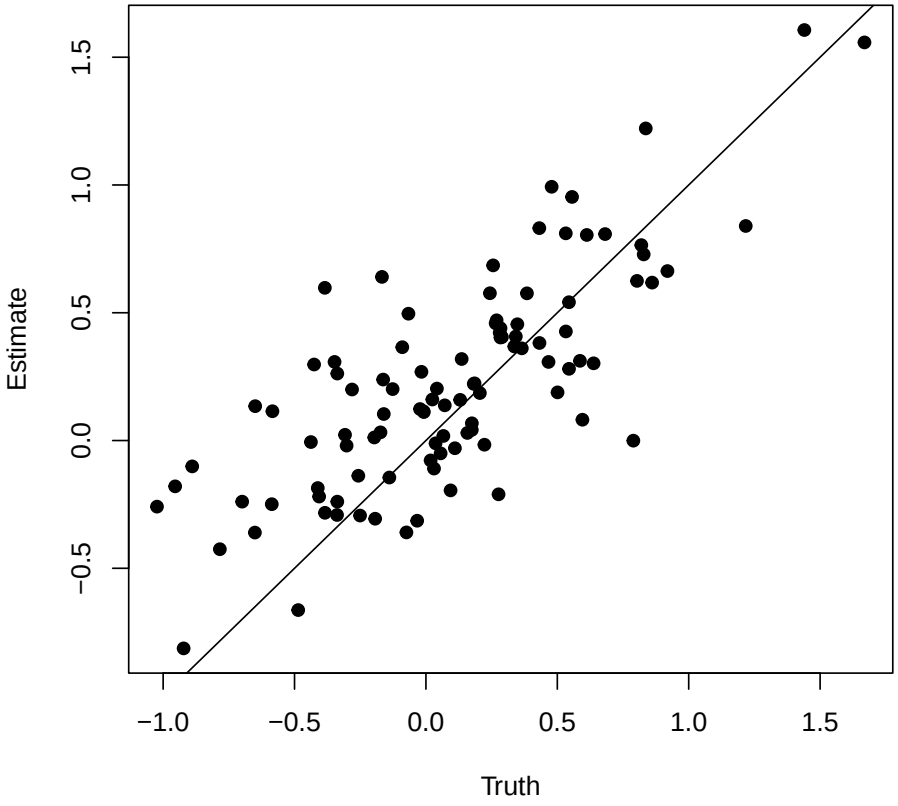
We can also extract the actual random effect values using the `randomTerms` function. We'll extract the values for the abundance model:

```
> head(randomTerms(mod_obs, "state"))
  Model  Groups        Name Level   Estimate        SE       lower
1   lam site_id (Intercept)     1  0.2407735 0.4322309 -0.60638340
2   lam site_id (Intercept)    10 -0.1444321 0.3024829 -0.73728763
3   lam site_id (Intercept)   100 -0.9993900 0.3040704 -1.59535695
4   lam site_id (Intercept)    11  0.4993507 0.2685825 -0.02706126
5   lam site_id (Intercept)    12 -0.2024980 0.4142401 -1.01439354
6   lam site_id (Intercept)    13 -0.4688996 0.3900390 -1.23336200
       upper
1  1.0879304
2  0.4484235
3 -0.4034231
4  1.0257627
```

```
5  0.6093976
6  0.2955629
```

Note the they are sorted incorrectly because `site_id` in this example, while numeric, is a factor so R sorts it like a character. Also note that these values are just the random part of the intercept - to get the complete intercept for each grouping level, we must add the mean intercept value (in this case 0.186). We can compare these estimates to the true values of the random intercepts.

```
> ran <- randomTerms(mod_obs, "state")
> ran <- ran[order(as.numeric(ran$Level)),] # sort them correctly
> ints <- coef(mod_obs)[1] + ran$Estimate # Calculate the total intercept for each level
> plot(re_site, ints, xlab="Truth", ylab="Estimate", pch=19)
> abline(a=0, b=1)
```




We can also use `predict` on models with random effects. A new argument is available, `re.form`, which specifies if the random effect(s) should be included when calculating the predicted estimate. By default, `re.form=NULL` meaning they are included; to exclude them set `re.form=NA`. These values are a bit confusing but they match the way it is done in `lme4`.

```
> # with random effect
> head(predict(mod_obs, "det"))
  Predicted         SE     lower     upper
1 0.4078631 0.04644771 0.3208759 0.5010342
2 0.5331650 0.05042090 0.4343354 0.6294585
3 0.4989963 0.04392033 0.4137838 0.5842672
4 0.5483388 0.05630166 0.4374296 0.6546455
5 0.4408197 0.04688357 0.3519180 0.5336856
6 0.5331650 0.05042090 0.4343354 0.6294585
```

```
> # without random effect
> head(predict(mod_obs, "det", re.form=NA))
  Predicted         SE     lower     upper
1 0.4907973 0.03341848 0.4258265 0.5560806
2 0.4907973 0.03341848 0.4258265 0.5560806
3 0.4907973 0.03341848 0.4258265 0.5560806
4 0.4907973 0.03341848 0.4258265 0.5560806
5 0.4907973 0.03341848 0.4258265 0.5560806
6 0.4907973 0.03341848 0.4258265 0.5560806
```

In the latter case, all the estimates of $p$ are identical because there are no fixed covariates on $p$ in this model.

### 2.4 More complicated random effects structures

It is possible to include multiple random effects on a single parameter; for example to include both observer and site as random effects on $p$, the formula for $p$ would look like this:

```
> ~1 + (1|obs_id) + (1|site_id)
```

Additionally it is possible to have random slopes as well as intercepts. For example, to also have random slopes for the covariate `x` by site, the formula for abundance would be

```
> ~x + (1 + x || site_id)
```

The `||` (indicating no correlation estimated between the two random effects) is necessary instead of `|`, as `unmarked` does not support correlated random effects.

## 3 A note on model selection

As you can see above, `unmarked` returns an AIC value for models with random effects. This AIC value is calculated in the normal manner, with the number of parameters equal to the number of fixed parameters plus the number of random effects standard deviations. There isn't a consensus on how to calculate AIC for models with fixed and random effects. Thus, it is probably not a good idea to use AIC to compare between models with and without random effects, even though `unmarked` will allow you to do so with `fitList` and `modSel`.