Modeling Abundance with Multinomial Mixtures

Outline:

A – Multinomial observation data
Point count data with "removal" protocol and similar

B – Analysis in `unmarked` using `multinomPois` and `gmultmix()`

C – Bayesian analysis in WinBUGS

D – Class work session - double observer data

E – Capture-recapture data: a type of multinomial
e.g., Replicated trapping grids

F – Class work session

PART A: Multinomial Observation Models

Multinomial Sampling Protocols

Bird sampling protocols: Many yield a **multinomial** $\mathbf{y}_i$ of counts for site $i = 1, 2, \ldots, M$.

Examples:

1. Capture/recapture

- T capture periods yields $K = 2^T - 1$ capture-history frequencies, $\mathbf{y}_i = \{y_{i,k}\}$. e.g., 2 sample periods:

history	frequency	
10	$y_{i,1}$	
01	$y_{i,2}$	K=3 **observable**
11	$y_{i,3}$	
00	$y_{i,0}$	$\longleftarrow$ **unobserved**

$$y_{i,0} = N_i - \Sigma_{k=1}^{3} y_{i,k}$$

Sampling protocols, continued

2. Multiple observer counts

- Counts are capture history frequencies
- Events indexed by observer instead of period

3. "Removal" counts

- y_{ik} = # birds first observed in time interval t

4. Distance sampling

- y_{ik} = # birds observed in distance interval k

Hierarchical models

We want to adapt the hierarchical modeling framework from the N-mixture model to accommodate these multinomial sampling protocols.

Abundance model:

$$N_i \sim Poisson(\lambda_i)$$

Observation model:

$$\mathbf{y}_i \sim Multinom(\boldsymbol{\pi}_i; N_i)$$

- Model abundance effects on λ_i
- Model observation effects on $\boldsymbol{\pi}_i$

The Observation Model

Multinomial likelihood for data from a single site:

$$f(\mathbf{y}_i|N_i, \mathbf{p}) = \frac{N_i!}{\left\{\Pi_{k=1}^{K} y_{ik}!\right\}(N_i - y_{i.})!}\left\{\prod_k \pi_k^{y_{ik}}\right\}(1-\pi.)^{N_i - y_{i.}}$$

- k is an index to distance class, capture-history, time interval, etc...
- π_k is a function of various (detection) parameters, $\mathbf{p}$

Difference among various multinomial sampling protocols is manifest in the parameterization of detectability in π_k.

Example: Capture-recapture (2 periods)

$$\begin{aligned}
\pi_1 &= \Pr('10') = \ p_1(1-p_2) \\
\pi_2 &= \Pr('01') = \ (1-p_1)p_2 \\
\pi_3 &= \Pr('11') = \ p_1 p_2
\end{aligned}$$

Example: Removal (3 periods)

$$\begin{aligned}
\pi_1 &= \Pr('1xx') = \ p_1 \\
\pi_2 &= \Pr('01x') = \ (1-p_1)p_2 \\
\pi_3 &= \Pr('001') = \ (1-p_1)(1-p_2)p_3
\end{aligned}$$

Factors that influence p –

$$\text{logit}(p_{it}) = a_0 + a_1 x_{it}$$

e.g., x_{it} = time (of day), temp., effort

The Multinomial Mixture: Summary

1. Data model (the likelihood) –

$$f(\mathbf{y}_i|N_i, \mathbf{p}) \propto \frac{N_i!}{(N_i - y_{.k})!} \left\{ \prod_k \pi_k^{y_{ik}} \right\} (1 - \pi_.)^{N_i - y_{.k}} \tag{1}$$

2. Abundance model –

$$N_i \sim g(N; \theta) \tag{2}$$

- local abundance is a "random effect"

Inference for Multinomial Mixtures

Likelihood Analysis

1. Remove them from the likelihood by integration.

The **integrated likelihood**:

$$f(\mathbf{y}|\mathbf{p},\theta) = \prod_i \left\{ \sum_{N_i=0}^{\infty} MN(\mathbf{y}_i|N_i, \boldsymbol{\pi}(\mathbf{p}))g(N_i|\theta) \right\}$$

With **Poisson** prior, simplifies to:

$$f(\mathbf{y}|\mathbf{p},\theta) = \prod_i \prod_k Poi(\lambda_i \pi_k)$$

2. Be Bayesian and analyze the posterior distribution by MCMC.
[details omitted]

Doing it in `unmarked`

- `unmarked` has 2 main functions for fitting multinomial abundance models:

  `multinomPois` – Multinomial-Poisson mixture (computationally efficient)

  `gmultmix` – Generalized multinomial-mixture model: other abundance models and **open populations**

- Helper-functions for set-up:

`unmarkedFrameGMM` or `unmarkedFrameMPois`: requires `type = removal` or `type = double` but other types can be described.

Multinomial Mixture: Analysis of Removal Data

R exercise:

Analyze bird point count data collected with a "removal" protocol using the functions `multinomPois` and `multmix`.

DOWNLOAD `Rscript_multinomial.R` from here:

`https://sites.google.com/site/unmarkedinfo/home/workshops/patux2012/r-scripts`

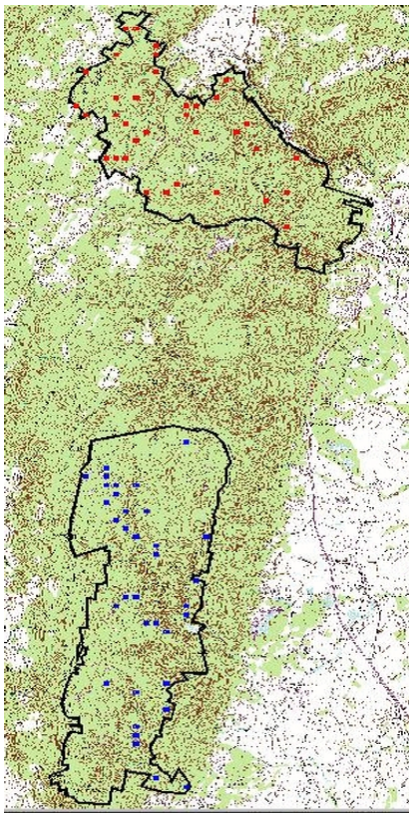
Example: Removal sampling protocol

Maryland forest comprising *Catoctin Mountain National Park*, Frederick Co. WMA, and various other land units.

Thanks to:

Deanna Dawson, USGS

Scott Bates, NPS



Sampling Details

- 70 point count locations
- Sampling protocol: "mental removal"
- 4 removal intervals
- **Habitat covariates**: Understory cover (UFC), basal area of large trees (TRBA)
- data set is provided with `unmarked`:

```
data(ovendata)
```

Multinomial example, continued

Removal counts:

$$\mathbf{y}_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4}); \quad i = 1, 2, \ldots, 70$$

Multinomial cell probabilities:

$$\begin{aligned}
\pi_1 &= p \\
\pi_2 &= (1-p)p \\
\pi_3 &= (1-p)^2 p \\
\pi_4 &= (1-p)^3 p
\end{aligned}$$

Poisson Abundance Model:

$$log(\lambda_i) = b_0 + b_1 UFC_i + b_2 BA_i$$

Ovenbird data –

	t=1	t=2	t=3	t=4
.	.	.	.	.
point 34	0	0	0	0
point 35	0	0	0	0
point 36	2	0	1	0
point 37	1	0	0	0
point 38	0	1	0	0
point 39	1	1	0	0
point 40	1	0	0	0
point 41	2	1	0	0
point 42	0	1	0	0
point 43	1	2	0	0
point 44	0	0	0	1
point 45	1	0	1	0
.	.	.	.	.
totals	49	16	5	7

←

Using multinomPois

```
##
## PART 1: Set-up the data
##

data(ovendata)  # built-in data

# note: every covariate is a "siteCovs" with a single primary sample, even if
# the covariate effects "p"

ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
    siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])),
    type = "removal")
```

Results

In class

Using multinomPois

```
##
## PART 2:Fit a bunch of models
##

fm0 <- multinomPois(~1 ~1,ovenFrame)
fm1 <- multinomPois(~ 1 ~ ufp + trba, ovenFrame)
fm2 <- multinomPois(~ 1 ~ ufp + trba + ufp*trba, ovenFrame)
fm3 <- multinomPois(~ ufp ~ ufp + trba, ovenFrame)
fm4 <- multinomPois(~ ufp ~ ufp + trba +ufp*trba, ovenFrame)

ms<- fitList(
"lam(.)p(.)"                              = fm0,
"lam(ufp+trba)p(.)"                       = fm1,
"lam(ufp+trba+ufp*trba)p(.)"              = fm2,
"lam(ufp+trba)p(ufp)"                     = fm3,
"lam(ufp+trba+ufp*trba)p(ufp)"            = fm4)

# Rank them by AIC
(ms1 <- modSel(ms))

# Table with everything you could possibly need
coef(ms)
```

Using `gmultmix`

- Allows negative binomial or zero-inflated abundance model
- Allows OPEN models (temporary emigration) [not covered right now]
- for closed population specify **`numPrimary=1`**
- Set-up:

```
ovenFrame <- unmarkedFrameGMM(ovendata.list$data,
    siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])),numPrimary=1,
    type = "removal")
```

- extra formula, order: λ, ϕ, p

```
fm0 <- gmultmix(~ 1 , ~ 1,  ~ 1 ,data=ovenFrame)
```

- Don't compare AIC results with **`multinomPois`** – likelihoods are scaled differently.

Ovenbird Results

```
data(ovendata)

ovenFrame <- unmarkedFrameGMM(ovendata.list$data,
    siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])),numPrimary=1,
    type = "removal")

fm0 <- gmultmix(~1 , ~1,  ~1 ,data=ovenFrame)
fm1 <- gmultmix(~ ufp+trba , ~1 , ~ 1, data=ovenFrame)
fm2 <- gmultmix(~ ufp+trba+ufp*trba , ~1 , ~ 1, data=ovenFrame)
# maybe p also depends on understory foliage?
fm3 <- gmultmix(~ ufp +trba, ~1 , ~ ufp,data= ovenFrame)
fm4 <- gmultmix(~ ufp + trba+ufp*trba,  ~1 , ~ ufp,data= ovenFrame)

fm0nb <- gmultmix(~1 , ~1,  ~1 ,mixture="NB",data=ovenFrame)
fm1nb <- gmultmix(~ ufp+trba , ~1 , ~ 1, mixture="NB",data=ovenFrame)
fm2nb <- gmultmix(~ ufp+trba+ufp*trba , ~1 , ~ 1, mixture="NB",data=ovenFrame)
# maybe p also depends on understory foliage?
fm3nb <- gmultmix(~ ufp +trba, ~1 , ~ ufp,,mixture="NB",data= ovenFrame)
fm4nb <- gmultmix(~ ufp + trba+ufp*trba,  ~1 , ~ ufp,mixture="NB",data= ovenFrame)

gms<- fitList(
"lam(.)p(.)"                                = fm0,
"lam(ufp+trba)p(.)"                         = fm1,
"lam(ufp+trba+ufp*trba)p(.)"                = fm2,
"lam(ufp+trba)p(ufp)"                       = fm3,
"lam(ufp+trba+ufp*trba)p(ufp)"              = fm4,
"NB,lam(.)p(.)"                                = fm0nb,
"NB,lam(ufp+trba)p(.)"                         = fm1nb,
"NB,lam(ufp+trba+ufp*trba)p(.)"                = fm2nb,
"NB,lam(ufp+trba)p(ufp)"                       = fm3nb,
"NB,lam(ufp+trba+ufp*trba)p(ufp)"              = fm4nb)
```

```
# Rank them by AIC
(gms1 <- modSel(gms))

# Table with everything you could possibly need
coef(gms1)
toExport <- as(gms1, "data.frame")

###
### Summary Results
###
> gms1
                                nPars    AIC delta AICwt cumltvWt
lam(ufp+trba)p(.)                   4 303.72  0.00 0.194     0.19
lam(.)p(.)                          2 303.86  0.14 0.180     0.37
lam(ufp+trba)p(ufp)                 5 304.21  0.49 0.152     0.53
lam(ufp+trba+ufp*trba)p(.)          5 304.75  1.03 0.116     0.64
lam(ufp+trba+ufp*trba)p(ufp)        6 305.30  1.58 0.088     0.73
NB,lam(ufp+trba)p(.)                5 305.72  2.00 0.071     0.80
NB,lam(.)p(.)                       3 305.78  2.06 0.069     0.87
NB,lam(ufp+trba)p(ufp)              6 306.21  2.49 0.056     0.93
NB,lam(ufp+trba+ufp*trba)p(.)       6 306.75  3.03 0.043     0.97
NB,lam(ufp+trba+ufp*trba)p(ufp)     7 307.32  3.60 0.032     1.00

##
##  Note: Dispersion parameter of NB is usually near the boundary 1/tau = 0
##  indicating no over-dispersion which is supported by the model-selection
##  results.
##  tau = exp(alpha)   [tau = "size"]
##  E[N] = lambda + (1/tau)*lambda^2
##
```

```
> print(coef(gms1),digits=2)
                          alpha(alpha) lambda(Int) lambda(trba) lambda(ufp) lambda(ufp:trba) p(Int) p(ufp)
lam(ufp+trba)p(.)                 NA        0.10        -0.17        0.10               NA   0.29     NA
lam(.)p(.)                        NA        0.13           NA          NA               NA   0.29     NA
lam(ufp+trba)p(ufp)               NA        0.11        -0.17        0.13               NA   0.34  -0.34
lam(ufp+trba+ufp*trba)p(.)        NA        0.15        -0.15        0.16             0.13   0.29     NA
lam(ufp+trba+ufp*trba)p(ufp)      NA        0.16        -0.15        0.19             0.13   0.34  -0.33
NB,lam(ufp+trba)p(.)             5.3        0.10        -0.17        0.10               NA   0.29     NA
NB,lam(.)p(.)                    2.9        0.13           NA          NA               NA   0.29     NA
NB,lam(ufp+trba)p(ufp)           7.0        0.11        -0.17        0.13               NA   0.34  -0.34
NB,lam(ufp+trba+ufp*trba)p(.)    8.1        0.15        -0.15        0.16             0.13   0.29     NA
NB,lam(ufp+trba+ufp*trba)p(ufp)  4.4        0.16        -0.15        0.19             0.13   0.35  -0.33
```

Conclusion:

- positive understory foliage effect on $E[N]$
- possibly negative effect on p
- negative effect of basal area of large trees on $E[N]$.
- No over-dispersion

Assessing Model Fit in `unmarked`

Bootstrap goodness-of-fit:

```
# need to create a function that computes fit statistics from the
# fit object in question

fitstats <- function(fm) {
    observed <- getY(fm@data)
    expected <- fitted(fm)
    resids <- residuals(fm)
    sse <- sum(resids^2,na.rm=TRUE)
    chisq <- sum((observed - expected)^2 / expected,na.rm=TRUE)
    freeTuke <- sum((sqrt(observed) - sqrt(expected))^2,na.rm=TRUE)
    out <- c(SSE=sse, Chisq=chisq, freemanTukey=freeTuke)
    return(out)
    }

gof <- parboot(fm1, fitstats, nsim=100, report=1)
```

Are any of these “tests” useful?

C. Bayesian Analysis in WinBUGS

- Bayesian analysis of multinomial observation models does not pose any novel technical difficulty.
- WinBUGS (and other BUGS) have a multinomial distribution function.
- In principle we just specify the multinomial/Poisson model directly:

```
y[i,]<-c(y[i,],NA)   # missing value for element "not captured"


###Then, in WinBUGS, do this:
y[i,] ~ dmulti( probs[i,], N[i] )
 N[i] ~ dpois(lambda[i])
```

- However, this construction doesn't work (in **WinBUGS**). Cannot have "random" sample size in multinomial distribution. (not so Binomial!)

C. Bayesian analysis in WinBUGS

We have 3 options:

(1) Multinomial/Poisson mixture has Poisson marginals. Use the Poisson marginal. (this is too easy!).

(2) Can use a "data augmentation" trick (Converse and Royle 2012) with individual-level encounter histories.

(3) Can express the model in terms of the conditional multinomial observation model. i.e., condition on n_i = number of individuals captured at site i:

$$
\begin{aligned}
y_i | n_i &\sim \text{Multinom}(\boldsymbol{\pi}_i^c, n_i) \\
n_i &\sim \text{Binomial}(1 - \pi_0, N_i) \\
N_i &\sim \text{Poisson}(\lambda_i)
\end{aligned}
$$

where $\pi_k^c = \pi_k/(1 - \pi_0)$

Bayesian analysis of removal model

```
cat("
model {

beta0 ~ dnorm(0,.1)
beta1 ~ dnorm(0,.1)
beta2 ~ dnorm(0,.1)
beta3 ~ dnorm(0,.1)

p0~dunif(0,1)

for(i in 1:nsites){
 p[i]<- p0   # could have covariates here
 mu[i,1]<- p[i]
 mu[i,2]<- p[i]*(1-p[i])
 mu[i,3]<- p[i]*(1-p[i])*(1-p[i])
 mu[i,4]<- p[i]*(1-p[i])*(1-p[i])*(1-p[i])
 pi0[i]<- 1 - mu[i,1]-mu[i,2]-mu[i,3]-mu[i,4]
 pcap[i]<-1-pi0[i]
 for(j in 1:4){
     muc[i,j]<-mu[i,j]/pcap[i]
 }
# 1. model part 1: the conditional multinomial
 y[i,1:4] ~ dmulti(muc[i,1:4],ncap[i])
# 2. model for the observed count of uniques
 ncap[i] ~ dbin(pcap[i],N[i])
# 3. abundance model
 N[i] ~ dpois(lambda[i])
 log(lambda[i])<- beta0 + beta1*X[i,1] + beta2*X[i,2] + beta3*X[i,2]*X[i,1]
 }
}
",fill=TRUE,file="model.txt")
```

Bayesian analysis of removal model

Set up the data, inits, parameters and send this off the **WinBUGS**:

```
y<- as.matrix(getY(ovenFrame))
nsites<-nrow(y)
ncap<-apply(y,1,sum)

ymax<-apply(y,1,sum)

data <- list(y=y,nsites=nsites,ncap=ncap,X=X)
inits <- function(){
  list (p0=runif(1),beta0=runif(1,-1,1),N=ymax+2 )
}

parameters <- c("N","p0","beta0","beta1","beta2","beta3")
nthin<-1
nc<-3
nb<-1000
ni<-2000

out <- bugs (data, inits, parameters, "model.txt", n.thin=nthin,n.chains=nc, n.burnin=nb,n.iter=ni,debug=TRUE)
```

Bayesian Analysis of Multinomial Mixtures

Results.... [in class]

Goodness-of-Fit using Bayesian p-values

- It is natural to think of overall "fit" as having two components: How well does the encounter model fit? How well does the abundance model fit?

- Can we evaluate them independently?

1. Fit of multinomial model conditional on n
2. Fit of model for n conditional on N_i – should be sensitive to wrong model for N_i

Bayesian p-values

- Define a statistic that quantifies fit (or lack thereof).

$$FIT = \sum(\sqrt{y} - \sqrt{E[y]})^2$$

- At each MCMC iteration compute this for the observed data, call this $FIT_{data}^{(i)}$
- obtain a new data set generated from the posterior distribution
- compute the fit statistic for this predicted data set $FIT_{pred}^{(i)}$
- Bayesian p-value = fraction of times $FIT_{data}^{(i)} > FIT_{pred}^{(i)}$.

- Implementation:

```
for(i in 1:N){

 ncap.fit[i] ~ dbin(pcap[i],N[i])
 y.fit[i,1:4] ~ dmulti(muc[i,1:4],ncap[i])  # note this is conditional on ncap[i]
 for(t in 1:4){
   e1[i,t]<- muc[i,t]*ncap[i]
   resid1[i,t]<- pow(pow(y[i,t],0.5)-pow(e1[i,t],0.5),2)
   resid1.fit[i,t]<- pow(pow(y.fit[i,t],0.5) - pow(e1[i,t],0.5),2)
   }
  e2[i]<- N[i]*lambda[i]
  resid2[i]<- pow(  pow(ncap[i],0.5) - pow(e2[i],0.5),2)
  resid2.fit[i]<- pow( pow(ncap.fit[i],0.5) - pow(e2[i],0.5),2)
}
ft1.data<- sum(resid1[,])
ft1.post<- sum(resid1.fit[,])

ft2.data<- sum(resid2[])
ft2.post<- sum(resid2.fit[])
```

Conducting the analysis:

```
parameters <- c("N","p0","beta0","beta1","beta2","beta3","ft1.data","ft1.post","ft2.data","ft2.post")
nthin<-1
nc<-3
nb<-1000
ni<-2000

out <- bugs (data, inits, parameters, "model.txt", n.thin=nthin,n.chains=nc, n.burnin=nb,n.iter=ni,debug=TRUE)

> mean(out$sims.list$ft1.data>out$sims.list$ft1.post)
[1] 0.292
> mean(out$sims.list$ft2.data>out$sims.list$ft2.post)
[1] 0.5386667
> save.image()
>
```

Model Selection in WinBUGS

Posterior model probabilities

- Expand model to include a set of binary indicator variables $w_k = 1$ if variable k is in the model.
- Model selection $\equiv$ estimating $\Pr(w_k = 1)$. (R&D Book, sec. 3.4.3)
- Example

$$
\begin{aligned}
log(\lambda_i) &= \beta_0 + w_1 * \beta_1 * x_{i1} + w_2 * \beta_2 * x_{i2} + w_1 * w_2 * w_3 * \beta_3 * x_{i1} * x_{i2} \\
w_1 &\sim Bern(0.5) \\
w_2 &\sim Bern(0.5) \\
w_3 &\sim Bern(0.5)
\end{aligned}
$$

Models characterized by $(w_1, w_2, w_1 w_2 w_3)$

- Estimate functions of w_k, e.g., $\Pr(w_k = 1)$
- Sensitivity to prior. Posterior model probabilities are sensitive to choice of prior distribution on β. See Link and Barker (2010).

In **WinBUGS**:

```
model {

beta0 ~ dnorm(0,.1)
beta1~ dnorm(0,.1)
beta2 ~ dnorm(0,.1)
beta3 ~ dnorm(0,.1)
w1~dbern(.5)
w2~dbern(.5)
w3~dbern(.5)

p0~dunif(0,1)

for(i in 1:nsites){
 p[i]<- p0   # could have covariates here
 mu[i,1]<- p[i]
 mu[i,2]<- p[i]*(1-p[i])
 mu[i,3]<- p[i]*(1-p[i])*(1-p[i])
 mu[i,4]<- p[i]*(1-p[i])*(1-p[i])*(1-p[i])
 pi0[i]<- 1 - mu[i,1]-mu[i,2]-mu[i,3]-mu[i,4]
 pcap[i]<-1-pi0[i]
 for(j in 1:4){
 muc[i,j]<-mu[i,j]/pcap[i]
 }

 y[i,1:4] ~ dmulti(muc[i,1:4],ncap[i])
 ncap[i] ~ dbin(pcap[i],N[i])
 N[i] ~ dpois(lambda[i])
 log(lambda[i])<- beta0 + w1*beta1*X[i,1] + w2*beta2*X[i,2] + w1*w2*w3*beta3*X[i,2]*X[i,1]
}

}
```

Post-processing to obtain model frequencies – combine the unique values of (w_1, w_2, w_3) into distinct models. i.e., $(1,0,0)$, $(0,1,0)$, $(1,1,0)$, etc.. Note: When w_3 represents an interaction we want to use $(w_1, w_2, w_1 w_2 w_3)$ so that the model has the interaction only if the main effects are present.

```
w1<-out$sims.list$w1
w2<-out$sims.list$w2
# new "w3" =1 only if the interaction is in the model, means w1 = 1 AND w2=1
w3<-out$sims.list$w3 * w1 * w2
mod<-paste(w1,w2,w3)

Prior: beta  ~ dnorm(0,.1)

> table(mod)
mod
0 0 0   0 1 0   1 0 0   1 1 0   1 1 1
 2176    517     300      6       1


Prior: beta ~ dnorm(0,.01)

0 0 0   0 1 0   1 0 0   1 1 0
 2760     154     78      8

Prior: beta ~ dnorm(0,.2)

0 0 0   0 1 0   1 0 0   1 1 0  1 1 1
 2006     637     340     16     1
```

Summary of analysis

- Model selection based on posterior model probabilities, when models represent different fixed covariates, is easy to accomplish using variable weights.
- The prior makes a difference, as it should.
- Note that some specific prior on β produces results similar to the "AIC weights" – so maybe we think that AIC is equivalent to using a specific prior for the model parameters.

D. Class work session

Double observer samples of Florida Manatees

- Thanks to Julien Martin, Florida DFW.
- Data are a subset of actual data and not proofed.
- $M = 324$ sample units "plots"
- Sampled by an airplane that circles over the plot with 2 observers. (3 passes each plot, here we analyze pass (1) only. Each pass is abt. 2 minutes.
- Visibility and seastate
  a. Sea State is determined by the Beaufort scale, from 0 (calm) to 4 (big waves)
  b. Visibility is indicated by a number from 1 to 4.
     * 1= Bottom is visible, manatees can clearly be seen
     * 2= Bottom visible but detail is unclear
     * 3= Although, water may be clear it is difficult to tell because bottom not visible due to depth
     * 4= Water is turbid (can see less than 1 m into it).
  c. long/lat?

Florida Manatee Data

Analyze these data.....

```
mtee<- read.csv("2011Abundance_AnalysisGroups_Andy_Workshop.csv")
mtee<-mtee[,c("Obs1Pass1","Obs2Pass1","Obs1and2Pass1","VisibilityID","SeaStateID",
             "CenterLong","CenterLat","Area")]
```

E. Capture-Recapture Protocols

- Almost all capture-recapture models imply a type of multinomial observation model.
- e.g., double observer is CR with $K = 2$.
- Special treatment here because it has broader applicability. Mist nets. Small mammal trap arrays.
- Can use `piFun` in tt unmarkedFrame to make the multinomial cell probabilities for basically *any* capture-recapture model and then fit N-mixture type models.
- Example here from:

  Royle, J.A., Kéry, M., Gautier, R. and Schmid, H. 2007. Hierarchical spatial models of abundance and occurrence from imperfect survey data. Ecological Monographs, 77:465-481.

The `piFun`

`piFun` – a function for converting a matrix of detection probabilities to multinomial cell probabilities. Converts a matrix of detection probabilities with $J =$ nreps columns into a matrix of cell probabilities with K columns. Each column corresponds to the probability of observing the encounter history k.

- Kind of important!
- `unmarkedFrameMPois` has default multinomial "types" – either `type = remove` or `type = double`. These declarations tell unmarked how to construct the multinomial cell probabilities using an internal function `piFun`.
- You can create your own types of multinomial models by specifying your own `piFun` instead of `type`.

Capture-recapture piFun

Here is a **piFun** to compute the multinomial cell probabilities when there were 3 sampling occasions: This function allows us to fit models M0, Mt, or models with covariates of p.

```
> crPiFun <- function(p) {
p1 <- p[,1]
p2 <- p[,2]
p3 <- p[,3]
cbind("001" = (1-p1) * (1-p2) * p3,
"010" = (1-p1) * p2 * (1-p3),
"011" = (1-p1) * p2 * p3,
"100" = p1 * (1-p2) * (1-p3),
"101" = p1 * (1-p2) * p3,
"110" = p1 * p2 * (1-p3),
"111" = p1 * p2 * p3)
}
```

To demonstrate how this works, imagine that we surveyed 2 sites and detection probability was constant (p = 0.4) among sites and survey occasions. The function converts these capture probabilities to multinomial cell probabilites:

```
> p <- matrix(0.4, 2, 3)
> crPiFun(p)
001 010 011 100 101 110 111
[1,] 0.144 0.144 0.096 0.144 0.096 0.096 0.064
[2,] 0.144 0.144 0.096 0.144 0.096 0.096 0.064
> rowSums(crPiFun(p))
[1] 0.784 0.784
```

obsToY: huh?

When providing a user-defined **piFun**, we also need to provide information about how to handle missing values. That is, if we have a missing value in a covariate, we need to know which values of y are affected (should not be included in the likelihood). In **unmarked**, this can be done by supplying a mapping-matrix called **obsToY** as an argument to **unmarkedFrameMPois**.

obsToY needs to be a matrix of zeros and ones with the number of rows equal to the number of columns for some **obsCov**, and the number columns equal to the number of columns in y. If **obsToY[j,k]=1**, then a missing value in **obsCov[i,j]** translates to a missing value in **y[i,k]**. For standard capture-recapture data, all elements of **obsToY** should *generally* be 1.

```
> o2y <- matrix(1, 3, 7)
```

So that if a covariate is missing in any sample, the encounter history is not contributing anything to the likelihood. Note: If in some cases a sample was not conducted, then care needs to be taken in how missing values are handled. (see MHB example).

Model M_t

We need to construct a covariate that references the time interval , which we call `intervalMat` below. We also provide a couple of site-specied covariates: the percent cover of woody vegetation and vegetation structure.

```
> library(unmarked)
> intervalMat <- matrix(c('1','2','3'), 50, 3, byrow=TRUE)
> class(alfl.H1) <- "matrix"
> umf.cr1 <- unmarkedFrameMPois(y=alfl.H1,
siteCovs=alfl.covs[,c("woody", "struct", "time.1")],
obsCovs=list(interval=intervalMat),
obsToY=o2y, piFun="crPiFun")
```

Now that we have our data formatted we can fit some models. The following correspond to model M0, model Mt, and a model with a continous covariate effect on p.

```
> M0 <- multinomPois(~1 ~1, umf.cr1)
> Mt <- multinomPois(~interval-1 ~1, umf.cr1)
> Mc <- multinomPois(~time.1 ~1, umf.cr1)
```

The first two models can be fit in other software programs. What is unique about unmarked is that we can also model variation in abundance and detection probability among sites (example to follow).

Summary of Capture-Recapture

- Writing a `piFun` and creating the `obsToY` object are the hardest parts of a capture-recapture analysis in `unmarked`.
- This is done automatically for removal models and double observer models, and we may add an option to do this automatically for capture-recapture data too, but hopefully the flexibility allowed by specifying user-defined functions is evident.

Application 2: Swiss Bird Data

Swiss survey of common breeding birds (abbrev "MHB").

- Observers walk route through 1 km quadrats, route length varies between about 3.5 km and 12 km.
- Observer walks same route 3 times (early, mid, late breeding season). Some high elev sites 2 times.
- Records all birds, what they're doing, where they are (on a map), and other information

Here we analyze "territory encounter histories"

Eurasian Jay data

Capture history frequencies for 10 quadrats for the *Eurasian Jay*:

```
> caphist[1:10,]
   100 010 001 110 101 011 111 10x 01x 11x
1    1   0   0   0   0   0   0  NA  NA  NA
2    0   0   1   0   0   0   0  NA  NA  NA
3    0   2   0   0   0   0   0  NA  NA  NA
4    1   0   0   0   0   0   0  NA  NA  NA
5    1   0   0   0   0   0   0  NA  NA  NA
6    1   0   0   2   0   1   0  NA  NA  NA
7    0   0   0   0   0   0   0  NA  NA  NA
8    0   0   0   1   1   0   0  NA  NA  NA
9    2   0   0   0   0   0   0  NA  NA  NA
10   1   1   3   2   2   4   0  NA  NA  NA
```

Covariates

Data for 15 of 237 1 km^2 quadrats:

|--abun covs-----|	|---- sampling/observation covariates ----------|								
elev	forest	length	day.1	day.2	day.3	dur.1	dur.2	dur.3	int.1	int.2	int.3
1300	32	6.1	35	58	75	260	270	290	42.6	44.3	47.5
1270	66	4.5	28	39	61	176	145	150	39.1	32.2	33.3
380	45	6.2	21	40	86	180	160	165	29.0	25.8	26.6
550	31	6.9	20	42	63	195	240	270	28.3	34.8	39.1
390	8	4.6	35	50	75	150	130	140	32.6	28.3	30.4
1380	78	3.7	54	75	99	160	160	150	43.2	43.2	40.5
530	37	5.7	30	50	76	225	210	225	39.5	36.8	39.5
1190	18	4.0	26	47	71	165	175	240	41.2	43.8	60.0
1490	54	5.5	47	62	77	300	325	321	54.5	59.1	58.4
920	6	4.3	20	40	61	170	90	98	39.5	20.9	22.8
620	3	3.4	17	46	61	270	271	215	79.4	79.7	63.2
540	27	4.2	15	45	66	180	240	180	42.9	57.1	42.9
820	56	5.3	37	60	90	290	315	295	54.7	59.4	55.7
1220	66	5.6	45	61	86	225	240	255	40.2	42.9	45.5
1180	43	5.1	39	60	83	325	365	350	63.7	71.6	68.6
730	40	6.2	23	45	58	390	391	406	62.9	63.1	65.5

Swiss Bird Data

Goals:

- What effects breeding abundance of this bird the species?
- Make a breeding abundance map of the species.
- Estimate total population size.

Hierarchical Model for Swiss Bird Data

Hierarchical model:

1. Model for quadrat population size:

$$\begin{aligned} N_i &\sim \text{Poisson}(\lambda_i) \\ log(\lambda_i) &= \beta_0 + covariates \end{aligned}$$

covariates = forest cover, elevation, route length

2. Multinomial observation model:

$$\begin{aligned} y &\in \{001, 010, 100, 011, 101, 110, 111\} \\ \Pr(y) &= \prod_t p_t^{y_t}(1-p_t)^{(1-y_t)} \end{aligned}$$

Detection covariates affect p_t

Variable route length

We use **inverse** of route-length as a covariate on abundance. Idea is the "exposed population", say N_i, for a quadrat is less than the actual population of the quadrat, say M_i, but as $L \to \infty$ (quadrat becomes saturated with sampling effort) then $N_i \to M_i$.

Quadrat population size model:

$$\begin{aligned} M_i &\sim Poisson(\lambda_i) \\ log(\lambda_i) &= \beta_0 + \beta_1 elev + \beta_2 forest \\ N_i|M_i &\sim Binomial(M_i, \phi(L_i)) \end{aligned}$$

where we model

$$\phi(L_i) = exp(-\beta_3/L_i)$$

Under this model the marginal distribution of N_i is

$$N_i \sim \text{Poisson}(\phi(L_i)\lambda_i)$$

Therefore

$$log(E[N_i]) = \beta_0 + \beta_1 elev + \beta_2 forest - \beta_3(1/L_i)$$

This means:

- Use inverse of route length to account for $N_i \rightarrow M_i$ as L_i increases.
- For making predictions we set $1/L_i = 0$ so that the prediction applies to saturation sampling effort.

Swiss Bird Data: Analysis

Part 1: Set-up the data for analysis

```
library("unmarked")
data<-source("MHB_jay.R")$value
```

```
> dim(data$caphist)
[1] 238  10
> data$caphist[1:4,]
  100 010 001 110 101 011 111 10x 01x 11x
1   1   0   0   0   0   0   0   0   0   0
2   0   0   1   0   0   0   0   0   0   0
3   0   2   0   0   0   0   0   0   0   0
4   1   0   0   0   0   0   0   0   0   0
```

```
caphist<-data$caphist
### If reps==2 then the first 7 records are missing
### if reps == 3 then the LAST 3 records are missing
reps<-ifelse(is.na(data$covinfo[,"dur3"]),2,3)
caphist[reps==2,1:7]<-NA
caphist[reps==3,8:10]<-NA
n<- nrow(caphist)

# need to set y[,1:7]=NA where T=2 and y[,8:10]=NA where T=3
# can pad covariates to not missing so obs don't get discarded?
```

```
crPiFun <- function(p) {
p1 <- p[,1]
p2 <- p[,2]
p3 <- p[,3]
cbind(
"100" = p1 * (1-p2) * (1-p3),
"010" = (1-p1) * p2 * (1-p3),
"001" = (1-p1) * (1-p2) * p3,
"110" = p1 * p2 * (1-p3),
"101" = p1 * (1-p2) * p3,
"011" = (1-p1) * p2 * p3,
"111" = p1 * p2 * p3,
"10x" = p1*(1-p2),
"01x" = (1-p1)*p2,
"11x" = p1*p2)
}

### get  covariates to use in model. Day of year and survey duration. SCALE THESE first
day=scale(data$covinfo[,c("date1","date2","date3")])
dur=scale(data$covinfo[,c("dur1","dur2","dur3")])
## Important:
## Then CHANGE the NA to any value. Why do this? If the plot was a T=2 replicate plot and we
## left day[,3] as missing then it would toss out that whole record. If we set day[,3]=0 it doesn't
## matter because day[,3] never enters the likelihood for that specific encounter history.
day[reps==2,3]<- 0
dur[reps==2,3]<- 0

obscovs<-list(dur=dur,day=day)

###
###
### Ok, we have to make the "obsToY" matrix....
###
### This method here will toss out a record if _any_ covariate is missing. Normally that
### is what you want, but not in the MHB with unbalanced replication
```

```
o2y <- matrix(1, 3, 10)

##
## to allow missing values in a more flexible way do this:
## o2y <- matrix(0, 3, 10)
# This way you can have NAs in your covariates and it wont toss out the
## whole record........ However, you need to make sure that if y(i,j) is real, then obscov(i,j) is also real.


covinfo<-data$covinfo
#
# These commands merge the landscape data with the observation locations
#
gridinfo<-read.csv("Swiss_landscape.csv")
ss<-paste(covinfo[,"x"],covinfo[,"y"],sep=".")
fore<-gridinfo[,"forest"]
names(fore)<-paste(gridinfo[,"x"],gridinfo[,"y"],sep=".")
forest<-fore[ss]
covinfo<-cbind(covinfo,forest=forest)

sitecovs<-covinfo[,c("elev","length","forest")]
sitecovs[,"elev"]<-scale(sitecovs[,"elev"])
sitecovs[,"forest"]<-scale(sitecovs[,"forest"])
# length NOT standardized
sitecovs[,"length"]<- 1/covinfo[,"length"]

caphist<-as.matrix(caphist)
mhb.umf <- unmarkedFrameMPois(y=caphist, siteCovs=as.data.frame(sitecovs),
              obsCovs=obscovs,obsToY=o2y, piFun="crPiFun")
```

Swiss Bird Data: Analysis

Part 2: Fit some models

```
caphist<-as.matrix(caphist)
mhb.umf <- unmarkedFrameMPois(y=caphist, siteCovs=as.data.frame(sitecovs),
              obsCovs=obscovs,obsToY=o2y, piFun="crPiFun")

fm1<- multinomPois(~1 ~1, mhb.umf)
fm2 <-multinomPois(~1 ~elev,mhb.umf)
fm3<- multinomPois(~1 ~forest,mhb.umf)
fm4<- multinomPois(~1 ~length,mhb.umf)
fm5<- multinomPois(~1 ~forest+elev,mhb.umf)
fm6<- multinomPois(~1 ~forest+length,mhb.umf)
fm7<- multinomPois(~1 ~elev+length,mhb.umf)
fm8<- multinomPois(~1 ~forest+elev+length,mhb.umf)
fm9<- multinomPois(~1 ~elev + I(elev^2),mhb.umf)
fm10<- multinomPois(~1 ~forest+elev+I(elev^2) + length,mhb.umf)

mspart1<- fitList(
"lam(.)p(.)"                        = fm1,
"lam(elev)p(.)"                     = fm2,
"lam(forest)p(.)"                   = fm3,
"lam(length)p(.)"                   = fm4,
"lam(forest+elev)p(.)"              = fm5,
"lam(forest+length)p(.)"            = fm6,
"lam(elev+length)p(.)"              = fm7,
"lam(forest+elev+length)p(.)"       = fm8,
"lam(elev + elev^2)p(.)"            = fm9,
"lam(forest+elev+elev^2+length)p(.)" = fm10)

#
# Rank them by AIC
(mspart1 <- modSel(mspart1))
```

```
                                   nPars     AIC  delta   AICwt cumltvWt
lam(forest+elev+elev^2+length)p(.)     6 1934.79   0.00 1.0e+00     1.00
lam(forest+elev+length)p(.)            5 1959.87  25.07 3.6e-06     1.00
lam(forest+elev)p(.)                   4 1966.51  31.72 1.3e-07     1.00
lam(elev + elev^2)p(.)                 4 1999.41  64.62 9.3e-15     1.00
lam(elev+length)p(.)                   4 2047.53 112.74 3.3e-25     1.00
lam(forest+length)p(.)                 4 2049.33 114.54 1.3e-25     1.00
lam(elev)p(.)                          3 2056.46 121.66 3.8e-27     1.00
lam(forest)p(.)                        3 2080.48 145.68 2.3e-32     1.00
lam(length)p(.)                        3 2120.05 185.25 5.9e-41     1.00
lam(.)p(.)                             2 2157.57 222.78 4.2e-49     1.00
>

# Table with everything you could possibly need
coef(mspart1)
toExport <- as(mspart1, "data.frame")
```

Swiss Bird Data: Goodness-of-Fit

```
fitstats <- function(fm) {      # NOTE: na.rm=TRUE !!!!!!
    observed <- getY(fm@data)
    expected <- fitted(fm)
    resids <- residuals(fm)
    sse <- sum(resids^2,na.rm=TRUE)
    chisq <- sum((observed - expected)^2 / expected,na.rm=TRUE)
    freeTuke <- sum((sqrt(observed) - sqrt(expected))^2,na.rm=TRUE)
    out <- c(SSE=sse, Chisq=chisq, freemanTukey=freeTuke)
    return(out)
    }

pb.mhb <- parboot(fm10, fitstats, nsim=100, report=1)

> pb.mhb

Call: parboot(object = fm10, statistic = fitstats, nsim = 100, report = 1)

Parametric Bootstrap Statistics:
               t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE           531           95.5             30.5       0.0099
Chisq        1702          231.3             81.6       0.0198
freemanTukey  413           29.0             13.1       0.0198

t_B quantiles:
               0% 2.5%  25%  50%  75% 97.5% 100%
SSE           379  384  416  433  454   495  549
Chisq        1336 1346 1413 1460 1507  1637 1862
freemanTukey  355  362  374  383  393   408  428

t0 = Original statistic compuated from data
t_B = Vector of bootstrap samples
```

Swiss Bird Data: Class Exercise

Can we come up with a better fitting model?

Analysis of the Swiss Bird Model

```
## Now we want to use these estimates to do a couple things
## Predict response of E[N] vs. elevation -- find optimal elevation
## make an abundance map over Switzerland

> min(siteCovs(mhb.umf)$elev)
[1] -1.442869
> max(siteCovs(mhb.umf)$elev)
[1] 2.425106
>

# remember length = 0 is saturation sampling because length = 1/L
new<- data.frame(elev=seq(-1.5,2.42,,50),forest=0,length=0)
pred<-predict(fm10,type="state",newdata=new,appendData=TRUE)
head(pred)

plot(Predicted ~ elev, pred,type="l",xlab="elevation (standardized)",
 ylab="Expected # territories",ylim=c(0,20))
lines(lower ~ elev, pred,type="l",col="red")
lines(upper ~ elev, pred,type="l",col="red")
```

```
# what is primo elevation for the willow tit?
# put on calculus hat.....
#   quadratic response:
#      y = a + b*x + c*x2
#   differentiate and set to 0:
#      dy/dx = b + 2*c*x = 0
#   solve
#      xopt = -b/(2*c)
#

b<- coef(fm10)[3]
c<- coef(fm10)[4]
elev.opt<- -b/(2*c)

elev.mean<- attr(siteCovs(mhb.umf)$elev,"scaled:center")
elev.sd  <- attr(siteCovs(mhb.umf)$elev,"scaled:scale")

elev.opt <- elev.opt*elev.sd + elev.mean
elev.opt
```

Spatial Prediction

```
###
### PART 3c:
### SPATIAL ANALYSIS/PREDICTION

landscape<-read.csv("Swiss_landscape.csv")
head(landscape)
## Note integer coordinates - row/column ids

gelev<- landscape[,"medel"]   # median elevation of quadrat
gforest<-landscape[,"forest"]
grid<-landscape[,c("x","y")]

# lets plot these variables to see how they look
#
# two options: (1) use my simpleton spatial.plot function
#              (2) stuff the data into a matrix and use image()
#
# grab utility functions including spatial.plot

source("utils.R")

par(mar=c(3,3,3,5),mfrow=c(2,1))
spatial.plot(grid,gelev)
spatial.plot(grid,gforest)
# this is cool
```

Spatial Prediction

```
# this is even cooler (could also use "raster" package but thats too easy):

# create a SQUARE matrix that will hold Switzerland grid attribute you wish to plot
# indexed by rows = "x coordinate" and columns = "y coordinate" as INTEGER
xmin <- min(grid[, 1])
xmax <- max(grid[, 1])
ymin <- min(grid[, 2])
ymax <- max(grid[, 2])
z <- matrix(NA, nrow = length(xmin:xmax), ncol = length(ymin:ymax))

##
## stuff attribute into matrix where matrix element [i,j] = coordinate [x,y] of
## the attribute to be plotted
## In this case we show an example of making an image plot of the ELEVATION attribute:
##
z[cbind(grid[, 1] - xmin + 1, grid[, 2] - ymin + 1)] <- gelev
ux <- unique(grid[, 1])
uy <- (sort(unique(grid[, 2])))

###
### make a new layer "z2" and set NA values (outside of switzerland) to -1
### for contouring
z2<-z
z2[is.na(z2)]<- -1

image(ux, uy, z, col = (terrain.colors(10)), axes = F)
image.scale(z, col = (terrain.colors(10)))
# make boundary by  putting a contour at 0
contour(ux, uy, z2, levels = c(0), add = T, labcex = 0.001)

####  Lets repeat that process but using the model predictions of E[N]
#
# To evaluate the predictions under the model we
```

```
# first we have to get the landscape covariates on the same scale as
# the model covariates. i.e., standardized.
#
elev.mean<- attr(siteCovs(mhb.umf)$elev,"scaled:center")
elev.sd <- attr(siteCovs(mhb.umf)$elev,"scaled:scale")
gelev<- (gelev - elev.mean)/elev.sd
forest.mean<- attr(siteCovs(mhb.umf)$forest,"scaled:center")
forest.sd <- attr(siteCovs(mhb.umf)$forest,"scaled:scale")
gforest<- (gforest - forest.mean)/forest.sd

#
# remember length = 0 is saturation sampling because length = 1/L
#
new<- data.frame(elev=gelev,forest=gforest,length=0)

## takes a long time
pred<-predict(fm10,type="state",newdata=new,appendData=TRUE)

### instead we have to do this the old-fashioned way:
### look at col names to figure order of parameters
###
par(mfrow=c(1,1))
betavec<-coef(fm10)[1:4]
Xg<-cbind(rep(1,length(gforest)),gforest,gelev,gelev*gelev)
pred<-exp(Xg%*%(betavec))

##
## stuff predictions into our raster
##
par(mar=c(3,3,3,6))
z[cbind(grid[, 1] - xmin + 1, grid[, 2] - ymin + 1)] <- pred
image(ux, uy, z, col = (terrain.colors(10)), axes = F)
image.scale(z, col = (terrain.colors(10)))
# make boundary by  putting a contour at 0
contour(ux, uy, z2, levels = c(0), add = T, labcex = 0.001)
#dev.off()
```

```
# lots of variation so try log-scale (or quantile coding)
z[cbind(grid[, 1] - xmin + 1, grid[, 2] - ymin + 1)] <- log(pred)
image(ux, uy, z, col = (terrain.colors(10)), axes = F)
image.scale(z, col = (terrain.colors(10)))
# make boundary by  putting a contour at 0
contour(ux, uy, z2, levels = c(0), add = T, labcex = 0.001)
```

Doing it in WinBUGS

Follow steps from previous analysis (removal example)

Space: The Final Frontier

Modification of abundance model:

$$\log(\lambda_i) = \beta_0 + \sum_j \beta_j v_{j,it} + z_i$$

$$\mathbf{z} \sim \text{SpatiallyDep}(\theta)$$

- Cannot analyze by likelihood methods
- Bayesian analysis in WinBUGS seems viable

Summary of Multinomial Mixture Models

- We can fit multinomial mixture models to a large number of multinomial protocols that arise in ecological sampling
- `unmarked` is unique in this capacity
- Can fit capture-recapture models in unmarked with "multiple populations". Can't do this anywhere else in any other capture-recapture software.
- Use the `piFun`, Luke
- A number of implementations in WinBUGS are possible including based on the conditional multinomial.