Multinomial N-mixture models

1. Multinomial observation data: Point count data with "removal", multiple observers or similar protocols
2. Analysis in `unmarked` using `multinomPois` and `gmultmix`
3. Custom multinomial models
   - Capture-recapture data
   - Chandler's flycatcher data.
4. Case study: territory mapping data

   mapping and prediction

   capture-recapture, also mist nets, replicated trapping grids

The multinomial distribution

Multivariate extension of the binomial: observation belongs in one of $H > 2$ classes

y = 'multinomial trial': Instead of (0,1) or (A,B), we have (A,B,C), etc.. Adding up multinomial trials produces a vector of frequencies (# class A, class B, class C…), denoted by: $\boldsymbol{y}$

We write:

$$\mathbf{y} \sim \text{Multinomial}(N, \boldsymbol{\pi}).$$

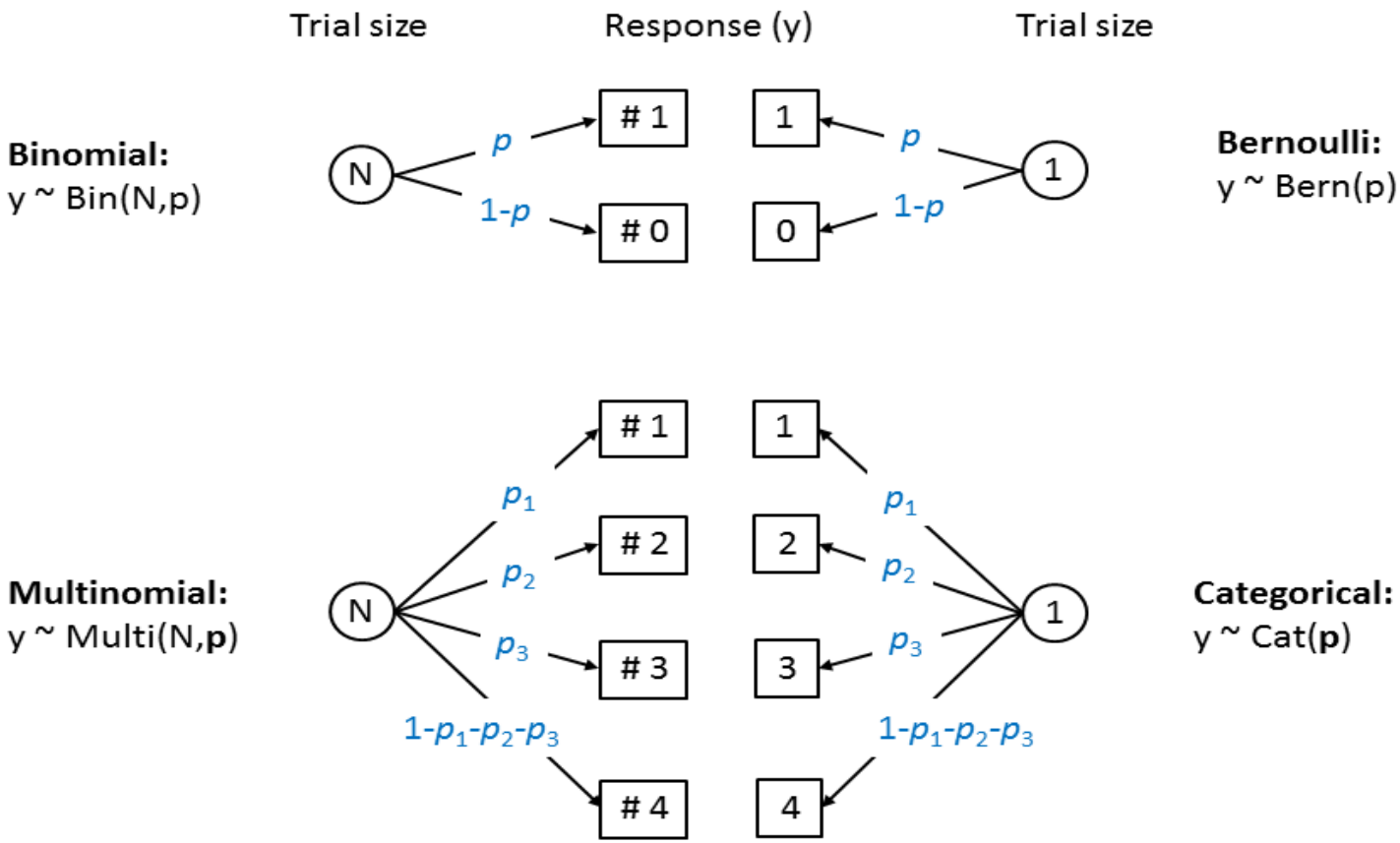
The multinomial distribution

PMF:

$$[\mathbf{y} \mid N, \boldsymbol{\pi}] \propto \frac{N!}{\prod y_h!} \left\{ \prod_{h=1}^{H} \pi_h^{y_h} \right\}$$

- Canonical distribution for “categorical frequency data”
- In the case of animal sampling, usually these are **unique encounter histories** (see next).
- Different protocols determine structure of the cell probabilities which depend on more basic “detection probability” parameters

The multinomial distribution



Multinomial N-mixture models

$$\mathbf{y}_i \mid N_i \sim \text{Multinomial}(N_i, \pi(p))$$

$$N_i \sim \text{Poisson}(\lambda)$$

- Form of multinomial cell probabilities $\pi(p)$ depends on sampling protocol used
- Covariates modeled on λ or p

Metapopulation sampling context

Spatially structured data

- Multinomial samples at each of i=1,2,…,M sites
- N_i = population size at site i (local population size)
- π_i = cell probabilities at site i

Multinomial observation models in ecological sampling

Many sampling protocols yield a type of multinomial count frequency data

- Example 1: **capture/recapture** or **multiple observer** K capture periods yields $H = 2^K - 1$ capture-history frequencies.

  e.g., 2 sample periods:

Encounter History	Frequency	Cell Probability	Function of p
10	y_1 or y_{10}	π_1	$p_1(1-p_2)$
01	y_2 or y_{01}	π_2	$(1-p_1)p_2$
11	y_3 or y_{11}	π_3	p_1p_2
00	y_4 or y_{00}	π_4	$(1-p_1)(1-p_2)$

Multinomial observation models in ecological sampling

- **Removal sampling:** y_{ik} = # birds first observed in time interval k

Encounter History	Frequency	Cell Probability	Function of p
1−−	y_1	π_1	p_1
01−	y_2	π_2	$(1-p_1)p_2$
001	y_3	π_3	$(1-p_1)(1-p_2)p_3$
000	y_4	π_4	$1-\pi_1-\pi_2-\pi_3$

- **Distance sampling:** y_{ik} = # birds observed in distance interval k

  [later]

Key idea 1: modeling p(t)

Factors that influence p

$$logit(p_{ik}) = \alpha_0 + \alpha_1 x_{ik}$$

x_{ik} = time of day, temperature, effort, etc..

Key idea 2: modeling N_i

- Local abundance varies across sites
- Random effects model:

$$N_i \sim g(\theta)$$

$$\log\big(E(N_i)\big) = \alpha + \beta * covariate[i]$$

Simulation practice

```
set.seed(2015)                          # Initialize RNG

# Simulate covariate values and local population size for each point
x <- rnorm(100)
N <- rpois(100, lambda=exp(-1 + 1*x) ) # Intercept and slope equal to 1
table(N)                                # Summarize
N
 0  1  2  3  4  5  6
72 17  6  1  2  1  1

# Define detection probabilities (p) for both observers
p1 <- 0.8
p2 <- 0.6

# Construct the multinomial cell probabilities (pi)
cellprobs <- c(p1*p2, p1*(1-p2), (1-p1)*p2, (1-p1)*(1-p2))

# Create a matrix to hold the data
y <- matrix(NA, nrow=100, ncol=4)
dimnames(y) <- list(1:100, c("11", "10", "01", "00"))

# Loop over sites and generate data with function rmultinom()
for(i in 1:100){
   y[i,] <- rmultinom(1, N[i], cellprobs)
}

# Remove 4th column ("not detected") and summarize results
y <- y[,-4]
apply(y, 2, sum)
11 10 01
23 17  6
```

Inference in multinomial mixtures

$$\mathbf{y}_i \mid N_i \sim \text{Multinomial}(N_i, \pi(p))$$

$$N_i \sim \text{Poisson}(\lambda)$$

- Latent variable or “random effect”
- Marginal likelihood (or integrated) – remove random effect from the conditional likelihood
- Bayesian analysis by MCMC: really easy also

Likelihood inference

$$f(\mathbf{y}_i|\mathbf{p},\theta) = \sum_{N_i=0}^{\infty} Multinomial(\mathbf{y}_i|N_i, \boldsymbol{\pi}(\mathbf{p}))g(N_i|\theta).$$

$$f(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M|\mathbf{p},\theta) = \prod_{i=1}^{M} f(\mathbf{y}_i|\mathbf{p},\theta).$$

Dual inference paradigm

- It's good to know both (see AHM book)
- But here we're just using "marginal likelihood" which is implemented in `unmarked`

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` and `unmarkedFrameMPois`
  - `type = removal` or `type = double` but other types can be described.

multinomPois

- General case, requires summation over the possible states of N_i

$$f(\mathbf{y}_i \mid \mathbf{p}, \theta) = \sum_{N_i=0}^{N_{max}} \text{Multinomial}(\mathbf{y}_i \mid N_i, \boldsymbol{\pi}(\mathbf{p})) g(N_i \mid \theta).$$

- Poisson case: This summation has an analytic solution:

$$f(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_3 \mid \mathbf{p}, \theta) = \prod_i \prod_h \text{Poisson}(\lambda_i \pi_h)$$

- Really fast!

Doing it in `unmarked`

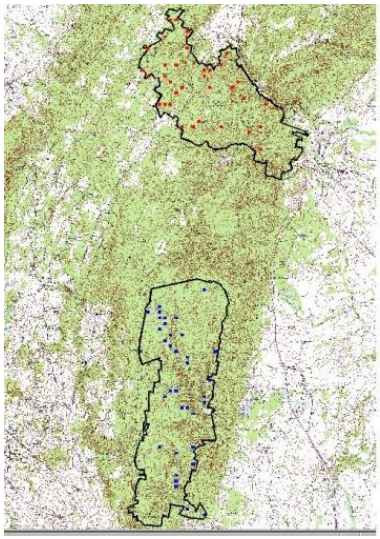
- Summary methods
  - `parboot`
  - `fitList`
  - `AIC, modSel`
  - `predict, fitted`
  - `raneff`
  - `bup`
  - `confint`

Worked example:

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

Deanna Dawson, USGS

Scott Bates, NPS



- 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)
```

```
data(ovendata)
ovendata.list$data[11:20,]   # Look at a snippet of data set
      [,1] [,2] [,3] [,4]
 [1,]    0    0    0    0
 [2,]    0    0    0    0
 [3,]    2    0    1    0
 [4,]    1    0    0    0
 [5,]    1    0    0    0
 [6,]    0    0    0    0
 [7,]    0    1    0    1
 [8,]    0    1    0    0
 [9,]    0    0    0    0
[10,]    0    0    0    0

apply(ovendata.list$data,2,sum)  # Removals in occasion 1-4
[1] 49 16  5  7
```

- The focus of the study had to do with effects of deer management on vegetation characteristics and hence bird populations and diversity (Zipkin et al. 2010).

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

$$\log(\lambda_i) = \beta_0 + \beta_1 \text{UFC}_i + \beta_2 \text{TRBA}_i$$

7.5.1 Setting up the data for analysis

Set things up for an analysis as an `unmarkedFrame`

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

Arguments:

- y = multinomial encounter frequencies
- type of multinomial observations ('removal' or 'double' are built-in)
- siteCovs is a matrix of covariates that vary by site
- obsCovs which is a list having 1 or more $nSite \times K$ matrices

7.5.2 Fitting models using function multinomPois

```
# Fit models: multinomPois
#       order of formulas: detection, abundance
fm0 <- multinomPois(~ 1 ~ 1, ovenFrame)
fm1 <- multinomPois(~ 1 ~ ufc, ovenFrame)
fm2 <- multinomPois(~ 1 ~ trba, ovenFrame)
fm3 <- multinomPois(~ 1 ~ ufc + trba, ovenFrame)
fm4 <- multinomPois(~ 1 ~ ufc + trba + ufc:trba, ovenFrame)
fm5 <- multinomPois(~ ufc ~ ufc + trba, ovenFrame)
fm6 <- multinomPois(~ ufc ~ ufc + trba + ufc:trba, ovenFrame)
```

Model selection

```
# Rank models by AIC
ms <- fitList(
"lam(.)p(.)"                                = fm0,
"lam(ufc)p(.)"                              = fm1,
"lam(trba)p(.)"                             = fm2,
"lam(ufc+trba)p(.)"                         = fm3,
"lam(ufc+trba+ufc:trba)p(.)"                = fm4,
"lam(ufc+trba)p(ufc)"                       = fm5,
"lam(ufc+trba+ufc:trba)p(ufc)"              = fm6)

(ms1 <- modSel(ms))
                            nPars    AIC delta AICwt cumltvWt
lam(trba)p(.)                   3 324.77  0.00 0.284     0.28
lam(ufc)p(.)                    3 325.73  0.96 0.176     0.46
lam(ufc+trba)p(.)               4 326.14  1.37 0.143     0.60
lam(.)p(.)                      2 326.28  1.51 0.134     0.74
lam(ufc+trba)p(ufc)             5 326.63  1.86 0.112     0.85
lam(ufc+trba+ufc:trba)p(.)      5 327.17  2.40 0.086     0.93
lam(ufc+trba+ufc:trba)p(ufc)    6 327.72  2.95 0.065     1.00
```

Parameter estimates

```
coef(ms1)[,1:4]  # Only first 4 columns shown
                              lambda(Int) lambda(trba) lambda(ufc) lambda(ufc:trba)
lam(trba)p(.)                   0.1062626   -0.2202827          NA               NA
lam(ufc)p(.)                    0.1134562           NA   0.1789146               NA
lam(ufc+trba)p(.)               0.1023681   -0.1708609   0.1002939               NA
lam(.)p(.)                      0.1296349           NA          NA               NA
lam(ufc+trba)p(ufc)             0.1069856   -0.1704169   0.1333443               NA
lam(ufc+trba+ufc:trba)p(.)      0.1547003   -0.1499694   0.1563209        0.1298563
lam(ufc+trba+ufc:trba)p(ufc)    0.1568600   -0.1503608   0.1854883        0.1264758
```

```
# Table with everything you could possibly need
 output <- as(ms1, "data.frame")
```

Summary of ovenbird

- Models containing the single variables TRBA and UFC or both are the top models
- Negative effect of basal area of large trees (TRBA) and a positive effect of understory cover (UFC) (ovenbird is an understory nesting and foraging species).
- The variables are highly (negatively) correlated which perhaps explains why the model with both variables only ranks 3rd in terms of AIC.

7.5.3 Fitting models using function `gmultmix`

Features of `gmultmix`:

- The negative binomial abundance model. The negative binomial parameterization used in `gmultmix` is by the mean, λ, and logarithm of the negative binomial 'size' parameter, say $\log(\tau)$. The variance is $\lambda + \lambda^2 / \tau$. Therefore, as $1/\tau \to 0$ (or $\tau \to \infty$), the negative binomial tends to the Poisson (i.e., no overdispersion is indicated).
- A type of *open* models (with temporary emigration) with an argument `numPrimary`, which is the number of primary sampling periods among which closure is not satisfied; see Chandler et al. (2011)
- Closed populations by specifying `numPrimary=1`.

7.5.3 Fitting models using function `gmultmix`

To set-up the data for analysis using `gmultmix`, we use the constructor function `unmarkedFrameGMM` as follows:

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


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

Fitting some models to the ovenbird data

```
# Fit Poisson models
fm1 <- gmultmix(~ ufc, ~ 1, ~  1, data = ovenFrame)
fm2 <- gmultmix(~ trba, ~ 1, ~ 1, data = ovenFrame)
fm3 <- gmultmix(~ ufc + trba, ~ 1, ~ 1, data = ovenFrame)
fm4 <- gmultmix(~ ufc + trba + ufc:trba, ~ 1, ~ 1, data = ovenFrame)
# Maybe p also depends on understory foliage?
fm5 <- gmultmix(~ ufc + trba, ~ 1, ~ ufc, data = ovenFrame)
fm6 <- gmultmix(~ ufc + trba + ufc:trba, ~ 1, ~ ufc, data = ovenFrame)

# Fit analogous NegBin models
fm0nb <- gmultmix(~ 1, ~ 1, ~ 1, mixture = "NB", data = ovenFrame)
fm1nb <- gmultmix(~ ufc, ~ 1, ~ 1, mixture = "NB", data = ovenFrame)
fm2nb <- gmultmix(~ trba, ~ 1, ~ 1, mixture = "NB", data = ovenFrame)
fm3nb <- gmultmix(~ ufc + trba , ~ 1, ~ 1, mixture = "NB", data =
ovenFrame)
fm4nb <- gmultmix(~ ufc + trba + ufc:trba, ~ 1, ~ 1, mixture = "NB",
    data = ovenFrame)
# maybe p also depends on understory foliage?
fm5nb <- gmultmix(~ ufc + trba, ~ 1, ~ ufc, mixture = "NB",
    data = ovenFrame)
fm6nb <- gmultmix(~ ufc + trba + ufc:trba, ~ 1, ~ ufc, mixture = "NB",
    data = ovenFrame)
```

Model selection and summary

```
# Rank models by AIC
gms <- fitList(
"lam(.)p(.)"                                    = fm0,
"lam(ufc)p(.)"                                  = fm1,
"lam(trba)p(.)"                                 = fm2,
"lam(ufc+trba)p(.)"                             = fm3,
"lam(ufc+trba+ufc:trba)p(.)"                    = fm4,
"lam(ufc+trba)p(ufc)"                           = fm5,
"lam(ufc+trba+ufc:trba)p(ufc)"                  = fm6,
"NB,lam(.)p(.)"                                 = fm0nb,
"NB,lam(ufc)p(.)"                               = fm1nb,
"NB,lam(trba)p(.)"                              = fm2nb,
"NB,lam(ufc+trba)p(.)"                          = fm3nb,
"NB,lam(ufc+trba+ufc:trba)p(.)"                 = fm4nb,
"NB,lam(ufc+trba)p(ufc)"                        = fm5nb,
"NB,lam(ufc+trba+ufc:trba)p(ufc)"               = fm6nb)

(gms1 <- modSel(gms))

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

Model selection and summary

```
# Summary results
gms1
                                  nPars    AIC delta AICwt cumltvWt
lam(trba)p(.)                         3 302.35  0.00 0.208     0.21
lam(ufp)p(.)                          3 303.32  0.96 0.128     0.34
lam(ufp+trba)p(.)                     4 303.72  1.37 0.105     0.44
lam(.)p(.)                            2 303.86  1.51 0.097     0.54
lam(ufp+trba)p(ufp)                   5 304.21  1.86 0.082     0.62
NB,lam(trba)p(.)                      4 304.35  2.00 0.076     0.70
lam(ufp+trba+ufp:trba)p(.)            5 304.75  2.40 0.063     0.76
lam(ufp+trba+ufp:trba)p(ufp)          6 305.30  2.95 0.047     0.81
NB,lam(ufp)p(.)                       4 305.31  2.95 0.047     0.85
NB,lam(ufp+trba)p(.)                  5 305.72  3.37 0.038     0.89
NB,lam(.)p(.)                         3 305.78  3.43 0.037     0.93
NB,lam(ufp+trba)p(ufp)                6 306.21  3.86 0.030     0.96
NB,lam(ufp+trba+ufp:trba)p(.)         6 306.75  4.40 0.023     0.98
NB,lam(ufp+trba+ufp:trba)p(ufp)       7 307.32  4.97 0.017     1.00
```

We see that the Poisson models are favored by AIC, i.e., no over-dispersion is indicated. This is supported by inspecting the summary results for the best negative binomial model `fm2nb`:

```
fm2nb

Call:
gmultmix(lambdaformula = ~trba, phiformula = ~1, pformula = ~1,
    data = ovenFrame, mixture = "NB")

Abundance:
            Estimate    SE      z P(>|z|)
(Intercept)    0.106 0.119  0.894   0.371
trba          -0.221 0.121 -1.831   0.067

Detection:
 Estimate    SE    z P(>|z|)
    0.288 0.233 1.24   0.216

Dispersion:
 Estimate   SE     z P(>|z|)
     4.57 16.9 0.271   0.786

AIC: 304.3497
```

Poisson model is adequate, no overdisperssion on local abundance. Note $\tau = \exp(4.57) \approx 97$

7.5.4 Assessing model fit in `unmarked`

- `parboot` function takes an R function defining the fit statistic(s), and an `unmarked` fit object, and carries-out a parametric bootstrap goodness-of-fit evaluation.
- `fitstats` (see the unmarked helpfile `?parboot`) : error sums-of-squares, the standard chi-square and the Freeman-Tukey statistic,

```
# 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)
```

7.5.4 Assessing model fit in `unmarked`

Best model object fm2, which is a Poisson model with TRBA on $\log(\lambda)$

```
set.seed(2015)
(gof <- parboot(fm2, fitstats, nsim = 1000, report = 1))
[... reporting output deleted ....]
Call: parboot(object = fm2, statistic = fitstats, nsim = 1000,
report = 1)

Parametric Bootstrap Statistics:
                t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE           72.4        -3.2696            12.05        0.595
Chisq        307.0        30.4393            28.95        0.126
freemanTukey  64.2        -0.0364             5.43        0.520

t_B quantiles:
              0% 2.5% 25% 50% 75% 97.5% 100%
SSE           46   55  67  75  83   102  124
Chisq        193  226 257 273 293   343  381
freemanTukey  48   54  60  64  68    74   79
```

These results indicate that the best model appears to fit the data reasonably well with the bootstrap p-value not being extreme (not so close to 0 or 1) for any of the three fit statistics.

7.7 Building custom multinomial models in `unmarked`

- By default `unmarked` can accommodate two types of multinomial sampling models: double observer and removal sampling.

- These options automatically create the pi function that converts encounter probability parameters to multinomial cell probabilities . Internally, `unmarked` has a special function (called the `piFun`) which builds the multinomial cell probabilities depending on the specified type of sampling protocol.

- The `piFun` maps “per sample” detection probabilities to multinomial cell probabilities.

- If you can build a `piFun` then unmarked will fit the multinomial model

- The `piFun` is passed as an argument to the `unmarkedFrame`

help file `?piFuns`.

```
# Removal model: capture probs for 5 sites, with 3 removal periods
(pRem <- matrix(0.5, nrow=5, ncol=3))
     [,1] [,2] [,3]
[1,]  0.5  0.5  0.5
[2,]  0.5  0.5  0.5
[3,]  0.5  0.5  0.5
[4,]  0.5  0.5  0.5
[5,]  0.5  0.5  0.5

removalPiFun(pRem)   # Multinomial cell probabilities for each site
     [,1] [,2]  [,3]
[1,]  0.5 0.25 0.125
[2,]  0.5 0.25 0.125
[3,]  0.5 0.25 0.125
[4,]  0.5 0.25 0.125
[5,]  0.5 0.25 0.125

# Double observer model: capture probs for 5 sites, with 2 observers
(pDouble <- matrix(c(0.8, 0.6), 5, 2,byrow=TRUE))
     [,1] [,2]
[1,]  0.8  0.6
[2,]  0.8  0.6
[3,]  0.8  0.6
[4,]  0.8  0.6
[5,]  0.8  0.6
doublePiFun(pDouble)  # Multinomial cell probabilities for each site
     [,1] [,2] [,3]
[1,] 0.32 0.12 0.48
[2,] 0.32 0.12 0.48
[3,] 0.32 0.12 0.48
[4,] 0.32 0.12 0.48
[5,] 0.32 0.12 0.48
>
```

Example

A `pi function` for removal sampling when the time intervals differ: 10-minute point count divided into three intervals of length 2, 3 and 5 minutes.

```
instRemPiFun <- function(p){
   M <- nrow(p)
   J <- ncol(p)
   pi <- matrix(NA, M, J)
   p[,1] <- pi[,1] <- 1 - (1 - p[,1])^2
   p[,2] <- 1 - (1 - p[,2])^3
   p[,3] <- 1 - (1 - p[,3])^5
   for(i in 2:J) {
      pi[,i] <- pi[, i - 1]/p[, i - 1] * (1 - p[, i - 1]) * p[, i]
   }
   return(pi)
}
```

The function expresses the detection probability for each time interval relative to a base per-minute detection probability and is defined as follows (see `?PiFuns`):

```
pRem<- matrix(0.5, 5,3)

instRemPiFun(pRem)
     [,1]    [,2]       [,3]
[1,] 0.75 0.21875 0.03027344
[2,] 0.75 0.21875 0.03027344
[3,] 0.75 0.21875 0.03027344
[4,] 0.75 0.21875 0.03027344
[5,] 0.75 0.21875 0.03027344
```

Other custom pi functions

```
crPiFun.Mb <- function(p) {
pNaive <- p[,1]
pWise <- p[,3]
 cbind("001" = (1 - pNaive) * (1 - pNaive) *      pNaive,
      "010" = (1 - pNaive) *      pNaive  * (1 - pWise),
       "011" = (1 - pNaive) *      pNaive  *      pWise,
     "100" =      pNaive  * (1 - pWise)  * (1 - pWise),
    "101" =      pNaive  * (1 - pWise)  *      pWise,
   "110" =      pNaive  *      pWise   * (1 - pWise),
   "111" =      pNaive  *      pWise   *      pWise)
}
```

Also “model Mh” (See AHM chapter 7)

7.8 Spatially Stratified Capture-Recapture Models

$$\mathbf{y_i} \mid N_i \sim \text{Multinomial}(N_i, \pi(p))$$

$$N_i \sim \text{Poisson}(\lambda)$$

- Includes classical capture-recapture
  - Small mammal studies at a M sites
  - Mist net studies of birds
  - Territory mapping
  - “mental” capture-recapture

Capture-recapture pi function

```
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)
}
```

With this we can fit some unique capture-recapture models that combine classical ideas (e.g., Mb, Mt) with N-mixture modeling ideas.

Example: Chandler's flycatcher data

```
alfl <- read.csv(system.file("csv", "alfl.csv",
package="unmarked"))
head(alfl, 5)
       id survey interval1 interval2 interval3
1 crick1_05      1         1         1         1
2 crick1_05      3         1         0         1
3   his1_05      1         0         1         1
4   his1_05      1         1         1         1
5   his1_05      2         0         1         1
```

There are 3 surveys with 3 intervals each. We use survey=1 only.

Example: Chandler's flycatcher data

```
alfl.covs <- read.csv(system.file("csv",
"alflCovs.csv",package="unmarked"),
 row.names=1)
head(alfl.covs)
          struct woody time.1 time.2 time.3 date.1 date.2 date.3
crick1_05   5.45  0.30   8.68   8.73   5.72      6     25     34
his1_05     4.75  0.05   9.43   7.40   7.58     20     32     54
hisw1_05   14.70  0.35   8.25   6.70   7.62     20     32     47
hisw2_05    5.05  0.30   7.77   6.23   7.17     20     32     47
kenc1_05    4.15  0.10   9.57   9.55   5.73      8     27     36
kenc2_05    9.75  0.40   9.10   9.12   9.12      8     27     36
```

Each row of the data matrix contains the covariate values for a given point count location, defined as follows:

- `struct` is a measure of vegetation structure
- `woody` is the percent cover of woody vegetation
- `time.x` is the time of day for each of the three sample occasions
- `date.x` is the day of each sample occasion.

Example: Chandler's flycatcher data

```
alfl$captureHistory <- paste(alfl$interval1, alfl$interval2,
alfl$interval3, sep="")
alfl$captureHistory <- factor(alfl$captureHistory,
    levels=c("001", "010", "011", "100", "101", "110",
       "111"))
alfl$id <- factor(alfl$id, levels=rownames(alfl.covs))

alfl.v1 <- alfl[alfl$survey==1,]
alfl.H1 <- table(alfl.v1$id, alfl.v1$captureHistory)
head(alfl.H1, 5)

            001 010 011 100 101 110 111
  crick1_05   0   0   0   0   0   0   1
  his1_05     0   0   1   0   0   0   1
  hisw1_05    0   0   0   0   0   0   0
  hisw2_05    0   0   0   0   0   0   1
  kenc1_05    0   0   0   0   0   0   0
```

Example: Chandler's flycatcher data

```
intervalMat <- matrix(c('1','2','3'), 50, 3, byrow=TRUE)
class(alfl.H1) <- "matrix"
o2y <- matrix(1, 3, 7)

umf.cr1 <- unmarkedFrameMPois(y=alfl.H1,
 siteCovs=alfl.covs[,c("woody", "struct", time.1")],
   obsCovs=list(interval=intervalMat), obsToY=o2y,
      piFun="crPiFun")

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

What????

Example: Chandler's flycatcher data

```
(M0.woody <- multinomPois(~ 1 ~ woody, umf.cr1))

Call:
multinomPois(formula = ~1 ~ woody, data = umf.cr1)

Abundance:
            Estimate    SE     z  P(>|z|)
(Intercept)   -0.962 0.325 -2.96 0.003059
woody          2.587 0.680  3.80 0.000143

Detection:
 Estimate    SE    z  P(>|z|)
     1.43 0.216 6.63 3.42e-11

AIC: 245.9301
```

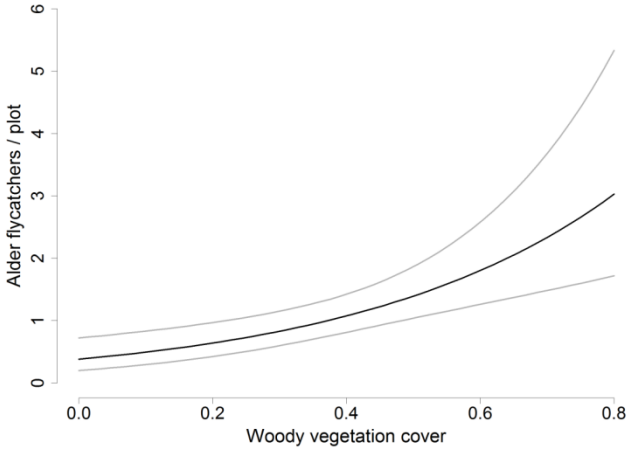
Example: Chandler's flycatcher data

```
fl <- modSel(fitList(M0, Mt, Mx, M0.woody))

          nPars   AIC delta   AICwt cumltvWt
M0.woody      3 245.93  0.00 0.99634     1.00
M0            2 258.24 12.31 0.00212     1.00
Mx            3 259.67 13.74 0.00104     1.00
Mt            4 261.10 15.17 0.00051     1.00
```

Example: Chandler's flycatcher data

Response curve:

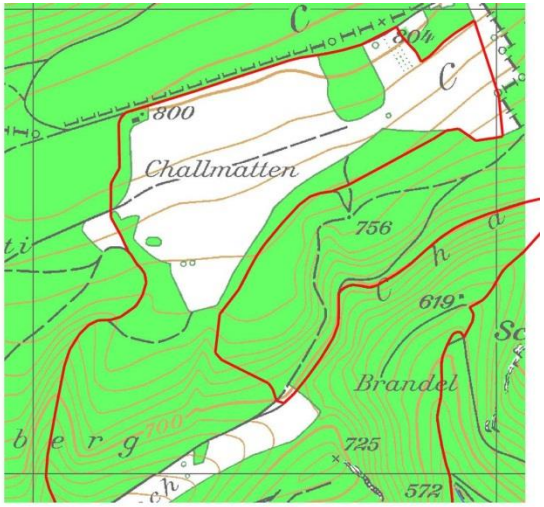
```
nd <- data.frame(woody=seq(0, 0.8, length=50))
E.abundance <- predict(M0.woody, type="state", newdata=nd,
appendData=TRUE)
plot(Predicted ~ woody, E.abundance, type="l", ylim=c(0, 6),
ylab="Alder flycatchers / plot", xlab="Woody vegetation cover",
frame = F)
lines(lower ~ woody, E.abundance, col=gray(0.7))
lines(upper ~ woody, E.abundance, col=gray(0.7))
```

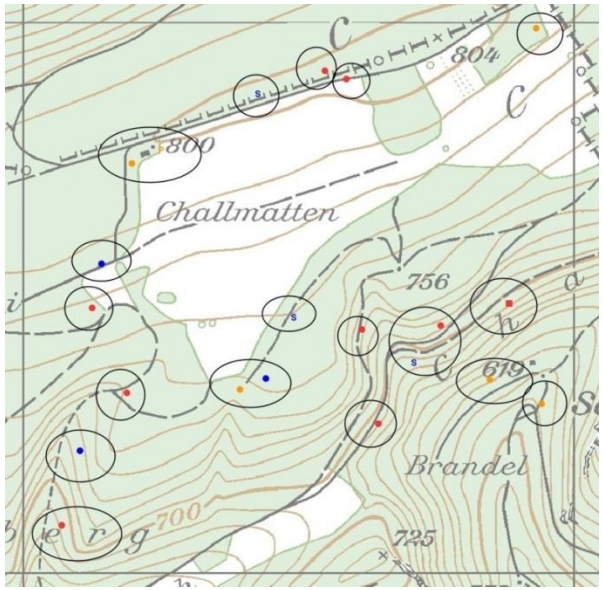


Class work session

- Is the ALFL a Poisson bird species?
- Fit some more covariate models
- Can you find a model that fits the data? Try both Poisson and NB models.

7.9 Case Study: Jays in the Swiss MHB





Territory encounter data:

```
data(jay)          # Load data
str(jay)           # Inspect data list
dim(jay$caphist)   # Look at detection history data
[1] 238  10

jay$caphist[1:4,]
  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
```

Mixed sampling protocol

Dual sampling protocol

- Mixed multinomial sample

$$\mathbf{y}_s = (y_{001,s}, y_{010,s}, y_{100,s}, y_{011,s}, y_{101,s}, y_{110,s}, y_{111,s}, y_{10x,s}, y_{01x,s}, y_{11x,s})$$

- One or the other set to NA

$$\mathbf{y}_s^{(1)} \sim \text{Multinomial}(\boldsymbol{\pi}_s(\mathbf{p}))$$
$$\mathbf{y}_s^{(2)} \sim \text{Multinomial}(\boldsymbol{\pi}_s(\mathbf{p}))$$

Swiss Jays

```
crPiFun <- function(p) {
   p1 <- p[,1] # Extract the columns of the p
matrix, one for
   p2 <- p[,2] #   each of J = 3 sample occasions
   p3 <- p[,3]
   cbind(      # define multinomial cell
probabilities:
      "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)
}
```

Covariates

```
|--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
```

- `elev`: average elevation of the quadrat in meters
- `forest`: forest cover (percent)
- `length`: length of the sample route through the quadrat in k
- `day.x`: integer day of the sample (day 1 = April 1)
- `dur.x`: duration of the survey (minutes)
- `int.x`: survey intensity, duration divided by length.

Parameterizations

- Intensity = Duration/Length
- Length = spatial coverage bias ???

Parameterizations

$$M_s \sim Poisson(\lambda_s)$$
$$log(\lambda_s) = \beta_0 + \beta_1 Elev + \beta_2 Forest$$
$$N_s \mid M_s \sim Binomial(M_s, \phi(L_s))$$

where we model

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

Under this model the marginal distribution of N_i is

$$N_s \sim \text{Poisson}(\phi(L_s)\lambda_s)$$

Therefore in our implementation of the model we assume

$$\log(E(N_s)) = \beta_0 + \beta_1\ Elev + \beta_2\ Forest - \beta_3(1/L_s)$$

Swiss Jays

```
# Grab the data objects
covinfo <- jay$covinfo
gridinfo <- jay$gridinfo
sitecovs <- jay$sitecovs
caphist <- jay$caphist

# Get observation covariates to use in model
# Day of year, sample intensity and survey duration.
# Standardize them.
day <- scale(covinfo[,c("date1", "date2", "date3")])
dur <- as.matrix(covinfo[,c("dur1", "dur2","dur3")])
dur[is.na(dur)] <- mean(dur, na.rm=TRUE) # WHY?
# Pad the 6 missing values

intensity <- dur / sitecovs[,"length"]
dur <- scale(dur)
intensity <- scale(intensity)
```

Swiss Jays

```
reps <- apply(!is.na(day), 1, sum)
day[reps==2,3] <- 0
dur[reps==2,3] <- 0
intensity[reps==2, 3] <- 0

# Store the observation covariates in a list
obscovs <- list(intensity = intensity, dur = dur, day = day)

# Standardize site covariates
sitecovs[,"elev"] <- scale(sitecovs[,"elev"])
sitecovs[,"forest"] <- scale(sitecovs[,"forest"])
# NOTE: length is NOT standardized, but over-written with its reciprocal
sitecovs[,"iLength"] <- 1 / sitecovs[,"length"]

# Create unmarkedFrame (need crPiFun above and unmarked loaded)
caphist <- as.matrix(caphist)
mhb.umf <- unmarkedFrameMPois(y=caphist,
siteCovs=as.data.frame(sitecovs),
   obsCovs=obscovs, obsToY=o2y, piFun="crPiFun")

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

??????

7.9.2. Fitting some models

```
# Fit a series of models
fm1 <- multinomPois(~1 ~1, mhb.umf)
fm2 <- multinomPois(~day ~1, mhb.umf)
fm3 <- multinomPois(~day + I(day^2) ~1, mhb.umf)
fm4 <- multinomPois(~intensity ~1, mhb.umf)
fm5 <- multinomPois(~intensity + I(intensity^2) ~1, mhb.umf)
fm6 <- multinomPois(~day + intensity ~1, mhb.umf)
fm7 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~1, mhb.umf)

# Assemble the models into a fitList and rank them by AIC
mspart1 <- fitList(
 "lam(.)p(.)" = fm1,
 "lam(.)p(day)" = fm2,
 "lam(.)p(day+day^2)" = fm3,
 "lam(.)p(intensity)" = fm4,
 "lam(.)p(intensity + intensity^2)" = fm5,
 "lam(.)p(day + rate)" = fm6,
 "lam(.)p(data + day^2 + intensity + intensity^2)" = fm7)

(mspart1 <- modSel(mspart1))
                                                  nPars     AIC delta   AICwt cumltvWt
lam(.)p(data + day^2 + intensity + intensity^2)       6 2057.62  0.00 4.9e-01     0.49
lam(.)p(day+day^2)                                    4 2057.73  0.11 4.6e-01     0.95
lam(.)p(day)                                          3 2063.01  5.39 3.3e-02     0.99
lam(.)p(day + rate)                                   4 2064.78  7.16 1.4e-02     1.00
lam(.)p(intensity + intensity^2)                      4 2147.18 89.56 1.7e-20     1.00
lam(.)p(intensity)                                    3 2155.25 97.63 3.1e-22     1.00
lam(.)p(.)                                            2 2157.57 99.95 9.7e-23     1.00
```

```
# Fit series of models
fm7 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~1, mhb.umf)
fm8 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~elev, mhb.umf)
fm9 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest, mhb.umf)
fm10 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~iLength, mhb.umf)
fm11 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest + elev, mhb.umf)
fm12 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest + iLength, mhb.umf)
fm13 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~elev + iLength, mhb.umf)
fm14 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest + elev + iLength, mhb.umf)
fm15 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~elev + I(elev^2), mhb.umf)
fm16 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest + elev + I(elev^2),
mhb.umf)
fm17 <- multinomPois(~day + I(day^2) + intensity + I(intensity^2) ~forest + elev + I(elev^2) +
iLength, mhb.umf)


# Assemble the models into a fitList
mspart2 <- fitList(
"lam(.)p(best)"                                = fm7,
"lam(elev)p(best)"                             = fm8,
"lam(forest)p(best)"                           = fm9,
"lam(length)p(best)"                           = fm10,
"lam(forest + elev)p(best)"                    = fm11,
"lam(forest + iLength)p(best)"                 = fm12,
"lam(elev + iLength)p(best)"                   = fm13,
"lam(forest + elev + length)p(best)"           = fm14,
"lam(elev + elev^2)p(best)"                    = fm15,
"lam(forest + elev + elev^2)p(best)"           = fm16,
"lam(forest + elev + elev^2 + iLength)p(best)"= fm17)
```

```
# Rank them by AIC
(mspart2 <- modSel(mspart2))
                                          nPars     AIC  delta   AICwt cumltvWt
lam(forest + elev + elev^2 + iLength)p(best)    10 1927.79   0.00 1.0e+00     1.00
lam(forest + elev + elev^2)p(best)               9 1942.53  14.74 6.3e-04     1.00
lam(forest + elev + iLength)p(best)              9 1947.20  19.41 6.1e-05     1.00
lam(forest + elev)p(best)                        8 1956.81  29.02 5.0e-07     1.00
lam(elev + elev^2)p(best)                        8 1991.30  63.51 1.6e-14     1.00
lam(forest + iLength)p(best)                     8 1992.53  64.75 8.7e-15     1.00
lam(forest)p(best)                               7 2012.67  84.88 3.7e-19     1.00
lam(elev + iLength)p(best)                       8 2018.26  90.47 2.3e-20     1.00
lam(elev)p(best)                                 7 2029.44 101.65 8.4e-23     1.00
lam(iLength)p(best)                              7 2039.10 111.31 6.7e-25     1.00
lam(.)p(best)                                    6 2057.62 129.83 6.4e-29     1.00
```

Best Poisson model (by AIC)

```
fm17

Call:
multinomPois(formula = ~day + I(day^2) + intensity + I(intensity^2) ~
    forest + elev + I(elev^2) + iLength, data = mhb.umf)

Abundance:
            Estimate     SE     z  P(>|z|)
(Intercept)    1.828 0.2505  7.30 2.92e-13
forest         0.370 0.0530  6.98 2.86e-12
elev          -0.674 0.0941 -7.16 7.89e-13
I(elev^2)     -0.480 0.1091 -4.40 1.09e-05
iLength       -4.392 1.1045 -3.98 6.98e-05

Detection:
                Estimate     SE       z  P(>|z|)
(Intercept)    -0.495411 0.1065 -4.6502 3.32e-06
day            -0.287148 0.0889 -3.2318 1.23e-03
I(day^2)        0.000859 0.0658  0.0131 9.90e-01
intensity       0.254052 0.0979  2.5949 9.46e-03
I(intensity^2) -0.047959 0.0798 -0.6009 5.48e-01

AIC: 1927.787
```

Use of gmultmix

Build the unmarkedFrame

```
mhb.umf2 <- unmarkedFrameGMM(y=caphist,
      numPrimary = 1,
      siteCovs=as.data.frame(sitecovs),
      obsCovs=obscovs, obsToY=o2y, piFun="crPiFun")
```

Fit some models:

```
fm1NB <- gmultmix(~1, ~1, ~1, mix = "NB", data = mhb.umf2)

fm17P <- gmultmix(~forest + elev + I(elev^2) + iLength,
~1, ~day + I(day^2) + intensity + I(intensity^2), mix =
"P",   data = mhb.umf2)

fm17NB <- gmultmix(~forest + elev + I(elev^2) + iLength,
~1, ~day + I(day^2) + intensity + I(intensity^2), mix =
"NB",   data = mhb.umf2)
```

7.9.3 Analysis of model fit

Two part fit assessment:

- We use the three fit statistics defined previously (sum-of-squared errors, chi-square, Freeman-Tukey;
- In addition we construct an analogous set of fit statistics but to evaluate the fit of the model for predicting , the observed number of individuals at quadrat . This measures the "spatial" model:

$$E(n_s) = \lambda_s(1 - \pi_0)$$

7.9.3 Analysis of model fit

```
# Define new fitstats function
fitstats2 <- function(fm) {
   observed <- getY(fm@data)
   expected <- fitted(fm)
   resids <- residuals(fm)
   n.obs<- apply(observed,1,sum,na.rm=TRUE)
   n.pred<- apply(expected,1,sum,na.rm=TRUE)
   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)
   freeTuke.n<- sum((sqrt(n.obs)-sqrt(n.pred))^2,na.rm=TRUE)
   sse.n <- sum( (n.obs -n.pred)^2,na.rm=TRUE)
   chisq.n <- sum((n.obs - n.pred)^2 / expected,na.rm=TRUE)

   out <- c(SSE=sse, Chisq=chisq, freemanTukey=freeTuke,
      SSE.n = sse.n, Chisq.n = chisq.n, freemanTukey.n=freeTuke.n)
   return(out)
}
```

These fit statistics measure the discrepancy of n_s from $E(n_s) = \lambda_s \pi_0$ and thus reflect mainly the "spatial part" of the model.

7.9.3 Analysis of model fit

```
(pb.mhb <- parboot(fm17, fitstats2, nsim=1000, report=1))

Call: parboot(object = fm17, statistic = fitstats2, nsim = 1000,
report = 1)

Parametric Bootstrap Statistics:
                  t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE              528       9.23e+01         2.89e+01      0.00200
Chisq           1704       2.36e+02         1.71e+02      0.02398
freemanTukey     409       2.98e+01         1.25e+01      0.00899
SSE.n           1512       2.13e+02         1.26e+02      0.05994
Chisq.n        44163      -5.21e+06         1.17e+08      0.09990
freemanTukey.n   203       3.69e+01         1.12e+01      0.00000

t_B quantiles:
                  0%  2.5%   25%   50%   75%  97.5%    100%
SSE              340   383   416   435   454    499 5.4e+02
Chisq           1258  1310  1398  1450  1501   1692 4.2e+03
freemanTukey     334   355   370   379   388    402 4.2e+02
SSE.n            879  1060  1209  1300  1381   1552 1.7e+03
Chisq.n        30267 32429 35406 37385 39780 164257 3.6e+09
freemanTukey.n   136   145   158   165   174    188 2.0e+02
```

7.9.3 Analysis of model fit

- In fact, the results suggest that perhaps this model does not adequately fit the data, having bootstrap p-values near 0 for each of the three fit statistics based on the encounter history frequencies. The fit is marginally better for predicting total observed territories but still not acceptable.

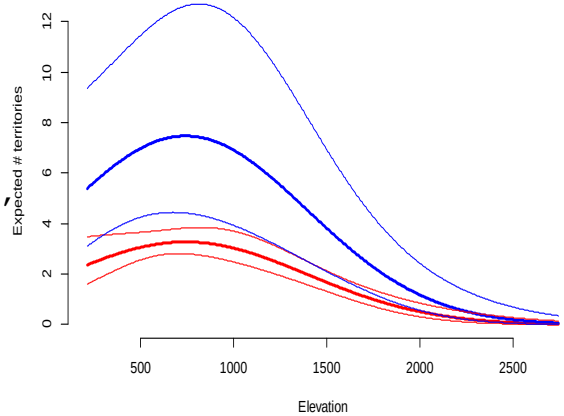
Fit of best NB model

```
(pb.mhbNB <- parboot(fm17NB, fitstats2, nsim=1000, report=1))

Call: parboot(object = fm17NB, statistic = fitstats2, nsim = 1000,
report = 1)

Parametric Bootstrap Statistics:
                  t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE              529       5.85e+01         3.89e+01       0.0779
Chisq           1706       1.71e+02         9.86e+01       0.0360
freemanTukey     409       2.19e+01         1.50e+01       0.0679
SSE.n           1544       1.96e+01         1.97e+02       0.4236
Chisq.n        43863      -2.08e+06         5.74e+07       0.3776
freemanTukey.n   203       3.38e+00         1.51e+01       0.3976

t_B quantiles:
                  0%  2.5%   25%   50%   75%  97.5%    100%
SSE              354   402   442   469   497    552 5.9e+02
Chisq           1327  1383  1475  1526  1582   1739 2.5e+03
freemanTukey     334   359   377   387   397    418 4.4e+02
SSE.n            998  1173  1390  1506  1655   1920 2.3e+03
Chisq.n        33123 36032 39928 42456 45755 216363 1.8e+09
freemanTukey.n   160   171   189   198   209    230 2.5e+02
```

7.9.4 Summary analyses of jay models

Making a response curve (elevation):

```
range(siteCovs(mhb.umf)$elev)
[1] -1.440050  2.429829

elev.mean <- attr(siteCovs(mhb.umf)$elev, "scaled:center")
elev.sd <- attr(siteCovs(mhb.umf)$elev, "scaled:scale")
elev.orig <- elev.sd*seq(-1.5, 2.42,,500)  + elev.mean


# Remember length = 0 is saturation sampling because length = 1/L
newL <- data.frame(elev = seq(-1.5,2.42,,500), elev.orig, forest = 0, iLength = 1/5.1)
# 'Low' prediction
newH <- data.frame(elev = seq(-1.5,2.42,,500), elev.orig, forest = 0, iLength = 0)            #
'High' prediction
predL <- predict(fm17NB, type="lambda", newdata=newL, appendData=TRUE)
predH <- predict(fm17NB, type="lambda", newdata=newH, appendData=TRUE)
head(cbind(low = predL[,1:2], high = predH[,1:2]))
  low.Predicted    low.SE high.Predicted  high.SE
1      2.366013 0.4657475       5.394389 1.518786
2      2.380799 0.4626372       5.428100 1.521632
3      2.395534 0.4594896       5.461694 1.524521
4      2.410214 0.4563058       5.495166 1.527454
5      2.424840 0.4530872       5.528511 1.530434
6      2.439407 0.4498348       5.561724 1.533460

plot(Predicted ~ elev.orig, predL, type="l", lwd = 3, xlab="Elevation",
 ylab="Expected # territories", ylim=c(0, 13), frame=F, col = "red")
points(Predicted ~ elev.orig, predH, type="l", lwd = 3, col = "blue")
matlines(elev.orig, predL[,3:4], lty = 1, lwd = 1, col = "red")
matlines(elev.orig, predH[,3:4], lty = 1, lwd = 1, col = "blue")
```



7.9.4 Summary analyses of jay models

$$y = a + b * x + c * x^2$$

$$\frac{dy}{dx} = \texttt{b + 2c*x}$$

$$x_{opt} = -\frac{b}{2c}$$

```
b <- coef(fm17NB)[3]
c <- coef(fm17NB)[4]
elev.opt <- -b / (2*c)

(elev.opt <- elev.opt*elev.sd + elev.mean)
lambda(elev)
```

```
    740.3788
```

Model averaged predictions

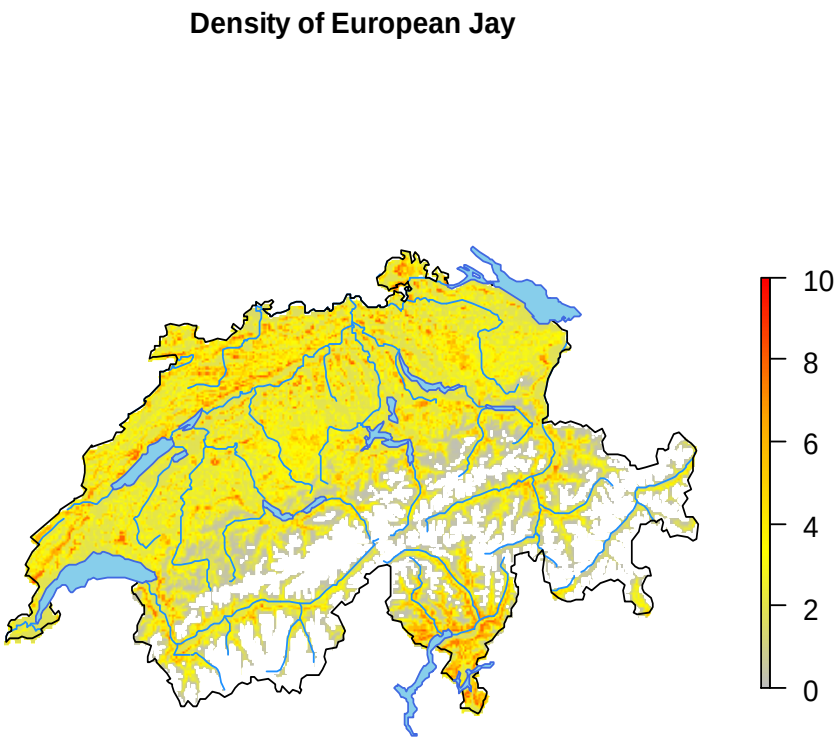
```
require(AICcmodavg)
model.list <- list(fm17NB) # candidate model list with single model
model.names <- c("AIC-best model")

# Compute model-averaged predictions of abundance for values of elevation, with
uncertainty (SE, CIs) adjusted for overdispersion (c.hat), with latter estimated
from bootstrapped Chisquare
pred.c.hatL <- modavgPred(cand.set = model.list, modnames = model.names, newdata
= newL, parm.type = "lambda", type = "response", c.hat = 1.11)

# Compare predictions and SE without and with c.hat adjustment
head(cbind(predL[1:2], pred.c.hatL[1:2]), 10)
Predicted        SE mod.avg.pred uncond.se
1   2.366013 0.4657475     2.366013 0.4906954
2   2.380799 0.4626372     2.380799 0.4874185
3   2.395534 0.4594896     2.395534 0.4841023
4   2.410214 0.4563058     2.410214 0.4807480
5   2.424840 0.4530872     2.424840 0.4773570
6   2.439407 0.4498348     2.439407 0.4739304
7   2.453915 0.4465501     2.453915 0.4704697
8   2.468361 0.4432342     2.468361 0.4669762
9   2.482743 0.4398885     2.482743 0.4634513
10  2.497058 0.4365144     2.497058 0.4598965
```

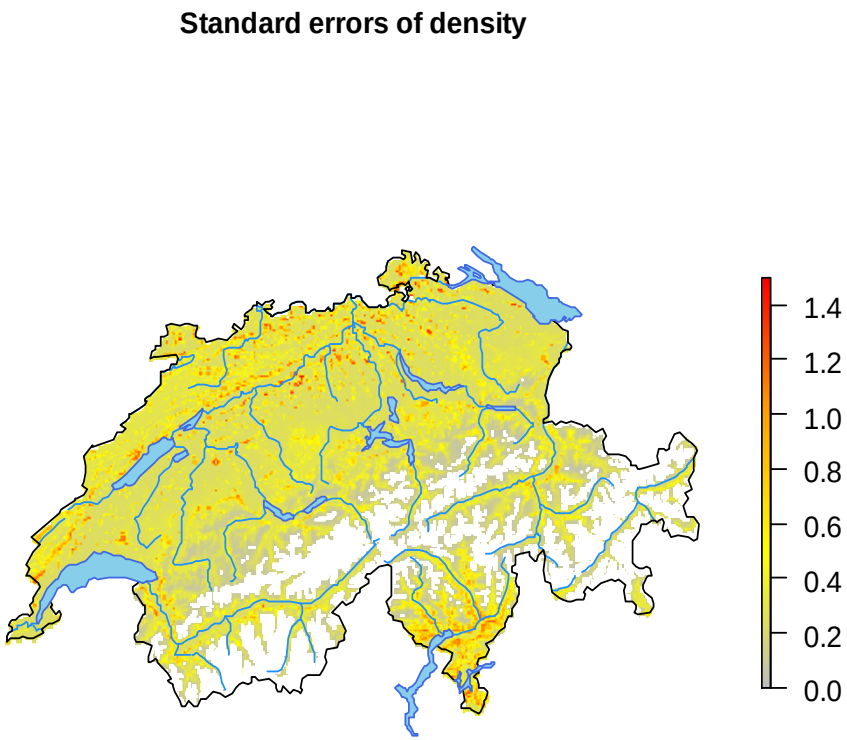
Spatial prediction

```
library(raster)
library(rgdal)

# Swiss landscape data and shape files
data(Switzerland)          # Load Swiss landscape data from unmarked
CH <- Switzerland          # this is for 'confoederatio helvetica'
head(CH)
gelev <- CH[,"elevation"] # Median elevation of quadrat
gforest <- CH[,"forest"]
grid <- CH[,c("x", "y")]
lakes <- readOGR(".", "lakes")
rivers <- readOGR(".", "rivers")
border <- readOGR(".", "border")
```

```
# Standardize elevation for all grid cells using the mean at sample plots
elev.mean <- attr(siteCovs(mhb.umf)$elev, "scaled:center")
elev.sd <- attr(siteCovs(mhb.umf)$elev, "scaled:scale")
gelev <- (gelev - elev.mean) / elev.sd

# Standardize forest cover also using the mean at sample plots
forest.mean <- attr(siteCovs(mhb.umf)$forest, "scaled:center")
foest.sd <- attr(siteCovs(mhb.umf)$forest, "scaled:scale")
gforest <- (gforest - forest.mean) / forest.sd

# Form predictions for Swiss landscape
newL <- data.frame(elev=gelev, forest=gforest, iLength=1/5.1)
newH <- data.frame(elev=gelev, forest=gforest, iLength=0)
pred.mhb.NB.Low <- predict(fm17NB, type="lambda", newdata=newL, appendData=T)
pred.mhb.NB.High <- predict(fm17NB, type="lambda", newdata=newH, appendData=T)

# Create rasters and mask for elevation (mask areas > 2250 m)
r1 <- rasterFromXYZ(cbind(x = CH$x, y = CH$y, z = pred.mhb.NB.Low[,1]))
r2 <- rasterFromXYZ(cbind(x = CH$x, y = CH$y, z = pred.mhb.NB.Low[,2]))
elev <- rasterFromXYZ(cbind(CH$x, CH$y, gelev))
elev[elev > 2250] <- NA
r1 <- mask(r1, elev)
r2 <- mask(r2, elev)

# Draw maps of jay density and standard error of density (Fig. 7-11)
par(mfrow = c(1,2), mar = c(1,2,3,5))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Density of European
Jay", zlim = c(0, 10))
plot(rivers, col = "dodgerblue", add = TRUE)
plot(border, col = "transparent", lwd = 1.5, add = TRUE)
plot(lakes, col = "skyblue", border = "royalblue", add = TRUE)
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Standard errors of
density", zlim = c(0, 1.5))
plot(rivers, col = "dodgerblue", add = TRUE)
plot(border, col = "transparent", lwd = 1.5, add = TRUE)
plot(lakes, col = "skyblue", border = "royalblue", add = TRUE)
```

Species Distribution Model (with SE)





Total population size

```
Nhat <- function(fm) {
   betavec <- coef(fm)[1:5]
   Xg <- cbind(rep(1, length(gforest)), gforest, gelev, gelev*gelev, 1/5.1)
   predLow <- as.numeric(exp(Xg%*%(betavec)))
   predHigh <- as.numeric(exp(Xg[,-5]%*%(betavec[-5])))
   out <- which(CH$water > 50 | CH$elevation > 2250)
   Nlow <- sum(predLow[-out])
   Nhigh <- sum(predHigh[-out])
   return(c(Nlow = Nlow, Nhigh = Nhigh))
}
set.seed(2015)
(pb.N <- parboot(fm17NB, Nhat, nsim=100, report=1))
```

```
[....monitoring output deleted....]

Parametric Bootstrap Statistics:
          t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
Nlow   88546          -96.6             5593        0.485
Nhigh 201879       -13019.3            59753        0.535

t_B quantiles:
          0%   2.5%    25%    50%    75%  97.5%   100%
Nlow   78052  80211  84744  88445  92168  99105 110939
Nhigh 111480 126430 168324 208866 248581 349855 408421

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

Summary

- A large number of important protocols yield multinomial observation vectors. Multinomial N-mixture models are really useful for such data.
- This is just a multivariate extension of the binomial N-mixture model.
- Spatially structured capture-recapture!!!
- `unmarked` has unique capabilities for analyzing these models. Can’t be found anywhere?