Distance-sampling analysis in unmarked

Richard Chandler

March 8, 2010

Abstract

Distance-sampling is a common wildlife sampling technique used to estimate population size or density. Describing how density varies spatially is often of equal interest; however, conventional methods of analysis do not allow for explicit modeling of both density and detection probability. The function distsamp implements the multinomial-Poisson mixture model of Royle et. al (2004), which was developed to overcome this limitation. This model requires that line or point transects are spatially replicated and that distance data are recorded in discrete intervals. This document describes how to format data, fit models, assess goodness-of-fit, and manipulate results.

1 Introduction

In distance sampling, the study design (line- or point-transect), distance class break points, transect lengths, and units of measurement need to be accounted for in the analysis. Unmarked uses an S4 class called unmarkedFrameDS to store data and metadata in a way that allows for easy data manipulation, summarization, and model specification. [1]

2 Importing, formatting, and summarizing data

The first step is to import the data into R. The simplest option is to use the read.csv function to import a .csv file that has been formatted so that each row represents a transect and columns describe either the number of individuals detected in each distance interval or transect-specific covariates. Alternatively, if data were not recorded in discrete distance intervals, a .csv file could be imported that contains a row for each individual detected and columns for the distances and transect names. This could then be converted to transect-level data using the function formatDistData. For example,

```
> library(unmarked)
> dists <- read.csv(system.file("csv", "distdata.csv",
     package = "unmarkedDev"))
> dists
     distance transect
1   0.6780015        a
2  18.3035080        a
3   7.0036037        a
4   2.4175874        a
5  12.8417904        b
6   2.6199117        b
7   5.2499713        b
8   9.5175094        b
9   5.4827487        c
10  9.3334616        c
11  1.3053622        c
12  2.1494693        c
13  3.6739826        d
14  2.8853327        d
15 12.3599871        d
16  6.6388658        d
17 10.7391135        e
18  0.1372242        e
```

```
19 20.2233830        e
20 18.6409361        e
21 13.0596460        f
22  0.4468151        f
23  2.0555385        f
24  7.9039541        f
> levels(dists$transect) <- c(levels(dists$transect), "g")
> levels(dists$transect)
[1] "a" "b" "c" "d" "e" "f" "g"
> yDat <- formatDistData(dists, distCol = "distance", transectNameCol = "transect",
    dist.breaks = c(0, 5, 10, 15, 20))
> yDat
  y.1 y.2 y.3 y.4
a   2   1   0   1
b   1   2   1   0
c   2   2   0   0
d   2   1   1   0
e   1   0   1   1
f   2   1   1   0
g   0   0   0   0
```

Here we have crated an object called yDat that contains counts for each transect (row) in each distance interval (columns). Note the method used to include transect "g", which was surveyd but where no individuals were detected. It is important that all survyed transects are included in the analysis.

Suppose there also exists transect-specific covariate data.

```
> (covs <- data.frame(canopyHt = c(5, 8, 3, 2, 4, 7, 5),
    habitat = c("A", "A", "A", "A", "B", "B", "B")))
  canopyHt habitat
1        5       A
2        8       A
3        3       A
4        2       A
5        4       B
6        7       B
7        5       B
```

These data can now organized along with the associated metadata using the function unmarkedFrameDS.

```
> umf <- unmarkedFrameDS(y = as.matrix(yDat), siteCovs = covs,
    survey = "line", dist.breaks = c(0, 5, 10, 15, 20),
    tlength = rep(100, 7), unitsIn = "m")
```

The arguments shown above indicate that the data were collected on seven line transects, each 100 meters long, and detections were tabulated into distance intervals defined by the dist.breaks cutpoints. It is important that both transect lengths and distance break points are provided in the same units specified by unitsIn.

We can look at these data, summarize them, and plot them usng a variety of methods.

```
> umf
Data frame representation of unmarkedFrame object.
  y.1 y.2 y.3 y.4 canopyHt habitat
a   2   1   0   1        5       A
b   1   2   1   0        8       A
c   2   2   0   0        3       A
d   2   1   1   0        2       A
e   1   0   1   1        4       B
f   2   1   1   0        7       B
g   0   0   0   0        5       B
> summary(umf)
```

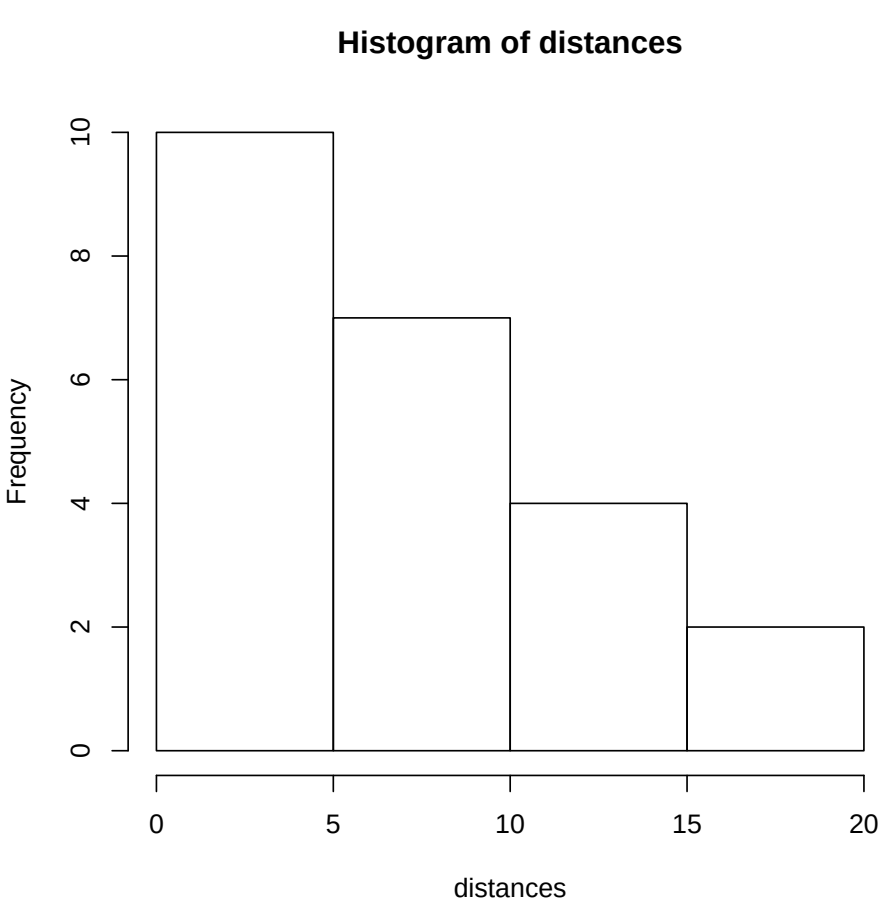
```
unmarkedFrameDS Object

line-transect survey design
Distance class cutpoints (m):  0 5 10 15 20
plot area information supplied? : FALSE

7 sites
Maximum number of distance classes per site: 4
Mean number of distance classes per site: 4
Sites with at least one detection: 6

Tabulation of y observations:
   0    1    2 <NA>
  11   11    6    0

Site-level covariates:
     canopyHt     habitat
 Min.   :2.000   A:4
 1st Qu.:3.500   B:3
 Median :5.000
 Mean   :4.857
 3rd Qu.:6.000
 Max.   :8.000
> hist(umf)
```



3 Model fitting, selection and evaluation

Now that we have put our data into an object of class unmarkedFrameDS, we are ready to fit some models with distsamp. The first argument is a formula which specifies the detection covariates followed by the density (or abundance) covariates. The only other required argument is the data, but several other optional arguments exist. By default, the half-normal detection function is used to model density in animals / ha. The detection function can be selected using the keyfun argument. The output can be changed from "density", to "abund" with the output argument. Note, however, that when line transect lenghs differ, they must be accounted for so abundance is actually animals per unit transect length. Whn modeling density, the output units can be changed from "ha" to "kmsq" using the unitsOut argument.

```
> hn_Null <- distsamp(~1 ~ 1, umf)
> haz_Null <- distsamp(~1 ~ 1, umf, keyfun = "hazard")
> (hn_Hab.Ht <- distsamp(~canopyHt ~ habitat, umf))
Call:
distsamp(formula = ~canopyHt ~ habitat, data = umf)

Density:
            Estimate    SE     z  P(>|z|)
(Intercept)    2.871 0.309  9.29 1.56e-20
habitatB      -0.556 0.457 -1.22 2.24e-01

Detection:
                 Estimate     SE     z  P(>|z|)
sigma(Intercept)   2.1203 0.4615 4.594 4.34e-06
sigmacanopyHt      0.0254 0.0897 0.283 7.77e-01

AIC: 63.1385
```

To rank models based upon AIC, we can use the function modSel after organzing them with the fitList function.

```
> mods <- fitList(hn_Null, haz_Null, hn_Hab.Ht)
> modSel(mods)
  n nPars    AIC deltaAIC   AICwt Rsq cumltvAICwt
1 7     2 60.724   0.0000 0.59851  NA     0.59851
2 7     3 62.702   1.9785 0.22256  NA     0.82107
3 7     4 63.138   2.4148 0.17893  NA     1.00000
```

Because parameter estimates are on the log-scale, we need to back-transform them to

References

[1] J. A. Royle. Generalized estimators of avian abundance from count survey data. *Animal Biodiversity and Conservation*, 27(1):375–386, 2004.