Introduction to R for distance sampling

Workshop, 26 August 2018

Centre for Research into Ecological and Environmental Modelling

1.3 Covariates in the detection function

In this practical, we illustrate fitting multiple covariate distance sampling (MCDS) models to point transect data using a bird survey in Hawaii; data on an abundant species, the Hawaii amakihi (*Hemignathus virens*) is used. This practical is based on the case study shown in Section 5.3.2 of Buckland *et al.* (2015) which duplicates the analysis presented in Marques *et al.* (2007). This set of data is included in 'Distance for Windows' as one of the Sample Projects: you can open this project (entitled amakihi.zip) in the Sample project directory underneath 'My Distance projects' directory residing under 'My Documents'. Here, we describe the analysis of these data using Distance in R.

Objectives of this practical

1. Introduce different types of plots to explore covariates
2. Add covariates to the detection function
3. Plot the detection functions.

Importing the data

Analysis begins by importing the data from a comma-delimited file (this file was created by copying the data from the amakihi Distance project and removing a few columns). Remember to copy the datafile to your R workspace or specify the directory where the data file is stored in the file name

```
# Import Amakihi data
amakihi <- read.csv(file="amakihi.csv")
```

Check that it has been imported correctly.

```
head(amakihi)
```

These data are made of of eight columns:

- Study.Area - name of the study area
- Region.Label - survey dates which are used as 'strata'

- Sample.Label - point transect identifier
- Effort - survey effort (which is always 1 because point transects used)
- distance - perpendicular distances
- OBS - initials of the observer
- MAS - minutes after sunrise
- HAS - hour after sunrise

The latter three columns are the covariates to be considered for possible inclusion into the detection function.

There a couple of records with missing distances and so can be deleted with the following command:

```r
# Get rid of missing values
amakihi <- amakihi[!is.na(amakihi$distance), ]
```

In this command,

- records in `amakihi` are selected using the square brackets `[]`
- `amakihi` is a data frame and so selection can be performed on either rows or columns i.e. `[rows, columns]`. In this case, the selection is performed on the rows (because the selection criteria is before the comma) and all columns will be retained
- the rows selected as those where the distances (stored in `amakihi$distance`) are not missing. The function `is.na` selects elements that are missing; the symbol `!` means 'not', and so `!is.na` selects elements that are not missing.

Exploratory data analysis

It is important to gain an understanding of the data prior to fitting detection functions (Buckland *et al.* 2015). With this in mind, preliminary analysis of distance sampling data involves:

- assessing the shape of the collected data,
- considering the level of truncation of distances, and
- exploring patterns in potential covariates.

Using the `summary` function provides a quick way to summarise of all the columns in the data set.

We begin by assessing the distribution of distances by plotting histograms with different number of bins and different truncation.

Plot a histogram of the distances with lots of bins and no truncation. In the command below, `seq(0, 260)` creates a sequence of numbers from 0 to 260, inclusive, at unit intervals (e.g. 0, 1, 2, 3, . . . , 260):

```r
par(mfrow=c(2,2))
hist(amakihi$distance, breaks=seq(0,260), main="",xlab = "Distance (m)")
```

Plot a histogram with lots of bins and truncation at 82.5 - this is the truncation distance that was used in Marques *et al.* (2007):

```r
hist(amakihi$distance[amakihi$distance<82.5], breaks=33, main="",xlab = "Distance (m)")
```

Plot a histogram with truncation at 82.5 but with fewer bins:

```r
hist(amakihi$distance[amakihi$distance<82.5], breaks=10, main="",xlab = "Distance (m)")
```

An alternative method of displaying the distribution of distances is a boxplot:

```r
boxplot(amakihi$distance, ylab="Distance (m)")
```

The components of the boxplot are:

- the thick black line indicates the median
- the lower limit of the box is the first quartile (25th percentile) and the upper limit is the third quartile (75th percentile)
- the height of the box is the interquartile range (75th - 25th quartiles)
- the whiskers extend to the most extreme points which are no more than 1.5 times the interquartile range.
- dots indicate 'outliers' if there are any, i.e. points beyond the range of the whiskers.

This format is probably not as useful as a histogram in a distance sampling context but boxplots can be useful to compare the distances for discrete groups in the data. Here we use boxplots to display the distribution of distances recorded by each observer and for each hour after sunrise. Note how the ~ symbol is used to define the groups.

Boxplots of distances by observer:

```r
# Boxplots by obs
boxplot(amakihi$distance~amakihi$OBS, xlab="Observer", ylab="Distance (m)")
```

Boxplot of distances for each hour after sunrise:

```r
# Boxplots by hour after sunrise
boxplot(amakihi$distance~amakihi$HAS, xlab="Hour", ylab="Distance (m)")
```

For minutes after sunrise (a continuous variable), we create a scatterplot of MAS (on the x-axis) against distances (on the y-axis). The plotting symbol (or character) is selected with the argument `pch`:

```r
# Plot of MAS vs distance (using dots)
plot(amakihi$MAS, amakihi$distance, xlab="Minutes after sunrise",
     ylab="Distance (m)", pch=20)
```

Adjusting the raw covariates

We would like to treat OBS and HAS as factor variables as in the original analysis; OBS is, by default, treated as a factor variable because it consists of characters rather than numbers. `HAS`, on the other hand, consists of numbers and so by default would be treated as a continuous variable (i.e. non-factor). That is fine if we want the effect of `HAS` to be monotonic (i.e. detectability either increases or decreases as a function of `HAS`). If we want `HAS` to have a non-linear effect on detectability, then we need to indicate to R to treat it as a factor as shown below.

```
# Convert HAS to a factor
amakihi$HAS <- factor(amakihi$HAS)
```

The next adjustment is to change the *reference* level of the *observer* and *hour* factor covariates - the only reason to do this is to get the estimated parameters in the detection function to match the parameters estimated by 'Distance for Windows'. By default R uses the first factor level but by using the `relevel` function, this can be changed:

```
# Set the reference level
amakihi$OBS <- relevel(amakihi$OBS, ref="TKP")
amakihi$HAS <- relevel(amakihi$HAS, ref="5")
```

One final adjustment, and more subtle, is a transformation of the continuous covariate, `MAS`. We are entertaining three possible covariates in our detection function: `OBS`, `HAS` and `MAS`. The first two variables, `OBS` and `HAS`, are both factor variables, and so, essentially, we can think of them as taking on values between 1 and 3 in the case of `OBS`, and 1 to 6 in the case of `HAS`. However, `MAS` can take on values from -18 (detections before sunrise) to >300 and the disparity in scales of measure between `MAS` and the other candidate covariates can lead to difficulties in the performance of the optimizer fitting the detection functions in R. The solution to the difficulty is to scale `MAS` such that it is on a scale (approx. 1 to 5) comparable with the other covariates.

Dividing all the `MAS` measurements by the standard deviation (function `sd`) of those measurements accomplishes the desired compaction in the range of the `MAS` covariate without changing the shape of the distribution of `MAS` values. The `na.rm=TRUE` argument ensures that any missing values are ignored.

```
# Rescale MAS by dividing by standard deviation
amakihi$MAS <- amakihi$MAS/sd(amakihi$MAS, na.rm=TRUE)
```

Check what this command has done by looking at a summary of the adjusted MAS:

```
summary(amakihi$MAS)
```

Candidate models

With three potential covariates, there are 8 possible combinations for including them in the detection function:

- No covariates
- OBS
- HAS
- MAS
- OBS + HAS
- OBS + MAS
- HAS + MAS
- OBS + HAS + MAS

Even without considering covariates there are a number of possible key function/adjustment term combinations and if all key function/covariate combinations are considered the number of potential models is large. Note that covariates are not allowed if a uniform key function is chosen and if covariate terms are included, adjustment terms are not allowed. Even with these restrictions, it is not best practice to take a scatter gun approach to detection function model fitting. Buckland *et al.* (2015) considered 13 combinations of key function/covariates. Here, we look at a subset of these as an illustration of how to incorporate covariates in the detection function.

If it is not already loaded, then first load the `Distance` package.

Fit a hazard rate model with no covariates or adjustment terms. By default, line transects are assumed and because our data are point transects, the argument `transect="point"` is specified:

```
hr.model0 <- ds(amakihi, transect="point", key="hr", truncation=82.5,
                adjustment=NULL, order=0)
```

The fitted model can be investigated using the `summary` function. Make a note of the AIC for this model.

```
summary(hr.model0)
```

Now fit a hazard rate model with OBS as a covariate in the detection function and make a note of the AIC. Has the AIC reduced by including a covariate?

```
hr.model1 <- ds(amakihi, transect="point", key="hr", formula=~OBS,
                truncation=82.5)
summary(hr.model1)
```

Fit a hazard rate model with OBS and HAS in the detection function:

```
hr.model2 <- ds(amakihi, transect="point", key="hr", formula=~OBS+HAS,
                truncation=82.5)
summary(hr.model2)
```

Try fitting other possible formula and decide which model is best in terms of AIC.

Plotting the detection functions

The detection functions can be investigated using the `plot` function as shown below. A few different plotting options are illustrated.

```
# Plot simple model
plot(hr.model0, nc=20, main="Simple model", pch=20)
```

```
# Plot model with OBS
plot(hr.model1, nc=10, main="Model with OBS", pch=1, pl.col=2)
```

What does the detection function look like for your selected model?

To see more sophisticated examples of plotting the detection function for the selected model, see the code accompanying Buckland *et al.* (2015) Hawaiian Amakihi case study.

References

Buckland ST, Rexstad EA, Marques TA and Oedekoven CS (2015) Distance Sampling: Methods and Applications. Springer 277 pp. ISBN: 978-3-319-19218-5 (Print) 978-3-319-19219-2 (Online)

Marques TA, Thomas L, Fancy SG and Buckland ST (2007) Improving estimates of bird density using multiple covariate distance sampling. The Auk 124:1229-1243