Introduction to distance sampling

Workshop, 21-23 August 2019

Centre for Research into Ecological and Environmental Modelling

Exercise 2. Line transect estimation using R

In this exercise, we use `R` (R Core Team 2018) and the `Distance` package (Miller 2017, Miller *et al.* 2019) to fit different detection function models to the duck nest data (introduced in Exercise 1) and estimate duck nest density and abundance.

1 Objectives

The aims of this exercise are to:

1. Load the Distance library
2. Import a data file
3. Fit a basic detection function using the `ds` function
4. Plot and examine a detection function
5. Assess goodness of fit of the detection function
6. Fit different detection function forms.

2 Survey data

As a reminder of the survey, 20 line transects, each of length 128.75 km, were searched out to a distance of 2.4 metres (Anderson and Pospahala 1970). Perpendicular distances to detected nests have been provided in a 'csv' text file in a basic format required by 'Distance' (more on this later). The columns in the file 'IntroDS_2.1.csv' are:

- Study.Area - this is the name of the study, Monte Vista NWR
- Region.Label - identifier of regions: in this case there is only one region and it is set to 'Default'
- Area - size of the study region (km^2): here the area is set to zero [1]
- Sample.Label - line transect identifier (numbered 1-20)
- Effort - length of the line transects (km)
- object - unique identifier for each duck nest identified
- distance - perpendicular distance (metres) to each duck nest.

The distances allow different key functions/adjustments to be fitted in the detection function model and, by including the transect lengths and area of the region, density and abundance can be estimated.

[1] The area of the refuge is 47.7 km^2 - this is needed to obtain abundance: for the purposes of this exercise, we are interested in fitting detection functions and density rather than abundance.

3 Getting started in R

Open `RStudio` (or `R` if you prefer working in the command window only). To let `R` know where to save the `R` work space and '.Rmd' files) set your 'working directory' - from the menu along the top of the `RStudio` window click on 'Session > Set Working Directory > Choose Directory' and select your chosen directory, for example 'C:/workshop'.

4 Importing the data

Read the data from a file into `R` with the following command - this command assumes that the `dsdata` package has been installed on your computer. A comma-delimited file 'IntroDS_2.1.csv' is in a directory that was created when that R package was installed.

```
# Import duck nest data
nestfile <- system.file("extdata", "IntroDS_2.1.csv", package = "dsdata")
nests <- read.csv(file=nestfile, header=TRUE)
```

To look at the first few rows of `nests` type the following command.

```
head(nests)
```

The object `nests` is a dataframe object made up of rows and columns. There is one row for each detected nest: use the function `nrow` to remind yourself how many detections there are:

```
nrow(nests)
```

5 Summarising the perpendicular distances

Create a numerical summary of the distances:

```
summary(nests$distance)
```

Similarly to plot a histogram of distances, the command is:

```
hist(nests$distance, xlab="Distance (m)")
```

We are now going to use the `Distance` package (Miller 2017) to fit a detection function to these data.

6 Using the `Distance` package

First, ensure that the `Distance` package (Miller 2017) has been installed if you haven't already installed it: to do this go to the `Packages` tab and click on `Install` and in the 'Packages' box type 'Distance'.

Once installed, the package can be loaded:

```
# Load package
library(Distance)
```

7 Fitting a simple detection function model with `ds`

Detection functions are fitted using the `ds` function and this function requires a data frame to have a column called `distance`. We have this in our `nests` data, therefore, we can simply supply the name of the data frame to the function as follows.

> A guaranteed way to produce incorrect results from your analysis is to mis-specify the units distances are measured. The `ds` function has an argument `convert.units` where the user provides a value to report density in proper units. Providing an incorrect value will result in estimates that are out by orders of magnitude.

Before fitting a model, the units of measure within the survey need to be reconciled. We can choose the units in which duck nest density is to be reported, we choose *square kilometres*. How to import this information to the `ds` function?

The answer is another function `convert_units`. Arguments to this function are - distance_units
- units of measure for perpendicular/radial distances - effort_units
- units of measure for effort (NULL for point transects) - area_units
- units of measure for the study area.

```
conversion.factor <- convert_units("meter", "kilometer", "square kilometer")
```

```
# Fit half-normal detection function, no adjustment terms
nest.hn <- ds(data=nests, key="hn", adjustment=NULL,
              convert.units=conversion.factor)
```

Details about the arguments for this function:

- `key="hn"`
  - fit a half-normal key detection function
- `adjustment=NULL`
  - do not include adjustment terms
- `convert.units=0.001`
  - required because, for this example, the perpendicular distances are in metres and the line transect lengths are in km - this argument converts the perpendicular distance measurements from metres to km.

As we have seen, on executing the `ds` command some information is provided to the screen reminding the user what model has been fitted and the associated AIC value. More information is supplied if we ask for a summary of the model as follows:

```
# Summarise model object
summary(nest.hn)
```

Can you match the information with the values you used in Exercise 1 - was your density estimate similar to the one obtained here?

To look at the fitted detection function, simply use the `plot` function:

```
plot(nest.hn)
```

The number of bins in the histogram can be changed by specifying the `nc` argument, for

example, to plot the histogram having 8 bins (as in Exercise 1) we can specify:

```
plot(nest.hn, nc=8)
```

The histogram should look like the one you drew in Exercise 1.

8 Goodness of fit

The usual tools for checking goodness of fit are available: the function `gof_ds` performs goodness of fits tests and plots a QQ-plot. In this command, 8 bins will be used for the chi-square goodness of fit test.

```
gof_ds(nest.hn, nc=8)
```

9 Specifying different detection functions

Different detection function forms and shapes, are specified by changing the `key` and `adjustment` arguments.

The different options available for `key` detection functions are:

- half normal (`key="hn"`) - this is the default
- hazard rate (`key="hr"`)
- uniform (`key="unif"`)

The different options available for adjustment terms are:

- no adjustment terms (`adjustment=NULL`)
- cosine (`adjustment="cos"`) - default
- Hermite polynomial (`adjustment="herm"`)
- Simple polynomial (`adjustment="poly"`)

For each model specified below, note down the AIC, density and 95% confidence interval and compare it to the model already fitted (i.e. half-normal with no adjustments). Which detection function model would you choose?

To fit a uniform key function with cosine adjustment terms, use the command:

```
nest.uf.cos <- ds(nests, key="unif", adjustment="cos",
                  convert.units=conversion.factor)
```

By default, AIC selection will be used to fit adjustment terms of up to order 5. Have any adjustment terms been selected?

To fit a hazard rate key function with Hermite polynomial adjustment terms, then use the command:

```
nest.hr.herm <- ds(nests, key="hr", adjustment="herm",
                   convert.units=conversion.factor)
```

10 References

Anderson DR and Pospahala RS (1970) Correction of bias in belt transect studies of immotile objects. The Journal of Wildlife Management 34:141-146. http://www.jstor.org/stable/3799501.

Miller DL (2017) Distance: Distance Sampling Detection Function and Abundance Estimation. R package version 0.9.7. https://CRAN.R-project.org/package=Distance

Miller DL, Rexstad E, Thomas L, Marshall L, Laake JL (2019) Distance Sampling in R. Journal of Statistical Software 89(1), 1-28. doi:10.18637/jss.v089.i01 http://doi.org/10.18637/jss.v089.i01.

R Core Team (2018) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Solution 2. Line transect estimation using R

Import and check the data.

```
# Import duck nest data
nestfile <- system.file("extdata", "IntroDS_2.1.csv", package = "dsdata")
nests <- read.csv(file=nestfile, header=TRUE)
# Check data OK
head(nests, n=3)
```

```
##        Study.Area Region.Label Area Sample.Label Effort object distance
## 1 Monte Vista NWR      Default    0            1 128.75      1     0.06
## 2 Monte Vista NWR      Default    0            1 128.75      2     0.07
## 3 Monte Vista NWR      Default    0            1 128.75      3     0.04
```

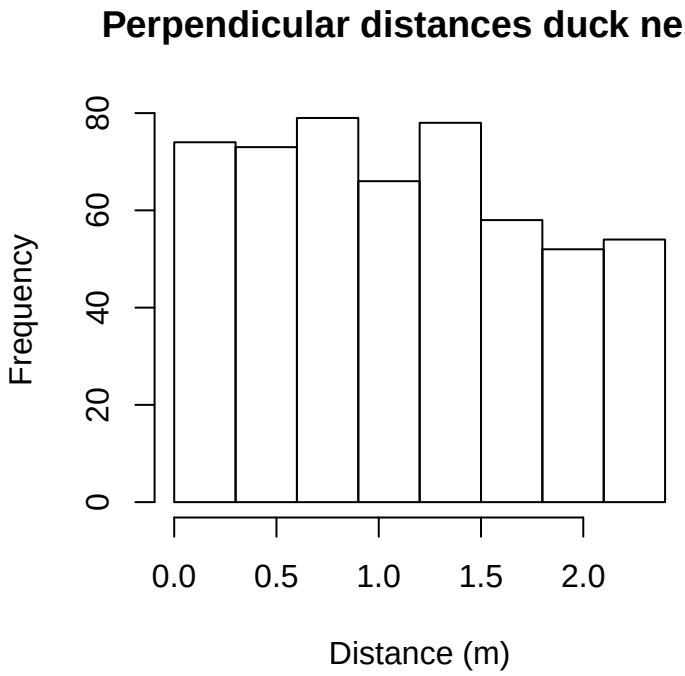
```
# How many observations (note: detections on all lines)
nrow(nests)
```

```
## [1] 534
```

```
# Summary of perp distances
summary(nests$distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   0.540   1.080   1.117   1.670   2.400
```

```
# Histogram
# Create 8 bins
brks <- seq(from=0, to=2.4, by=0.3)
hist(nests$distance, breaks=brks, xlab="Distance (m)",
     main="Perpendicular distances duck nests")
```



Fit the three models using proper units of distance measure..

The answer is another function `convert_units`. Arguments to this function are - distance_units
- units of measure for perpendicular/radial distances - effort_units
- units of measure for effort (NULL for point transects) - area_units
- units of measure for the study area.

```
# Load library
library(Distance)
conversion.factor <- convert_units("Meter", "Kilometer", "Square Kilometer")
# Model 1. Half-normal with no adjustments
nest.hn <- ds(nests, key="hn", adjustment=NULL,
              convert.units=conversion.factor)
# Summary
summary(nest.hn)
```

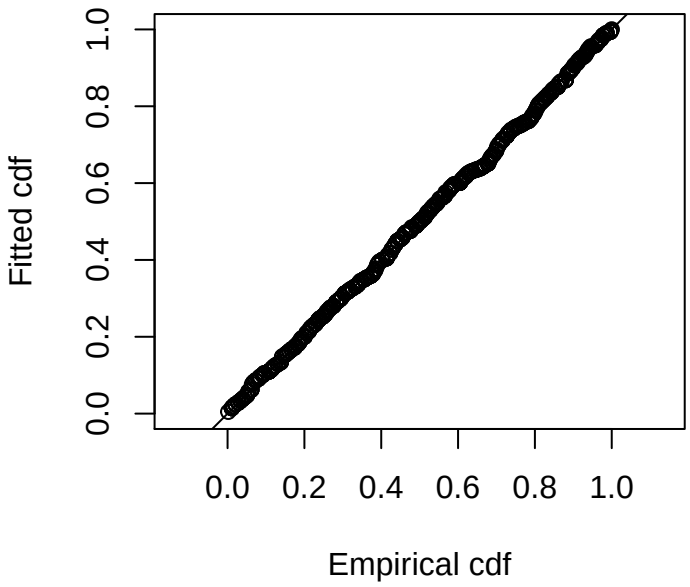
```
##
## Summary for distance analysis
## Number of observations :  534
## Distance range         :  0  -  2.4
##
## Model : Half-normal key function
## AIC   : 928.1338
##
## Detection function parameters
## Scale coefficient(s):
##               estimate        se
## (Intercept) 0.9328967 0.1703933
```

```
##
##                          Estimate          SE         CV
## Average p               0.8693482  0.03902053 0.04488481
## N in covered region 614.2533225 29.19683067 0.04753223
##
## Summary statistics:
##    Region  Area CoveredArea Effort   n  k        ER       se.ER      cv.ER
## 1 Default 12.36       12.36   2575 534 20 0.2073786 0.007970756 0.03843576
##
## Density:
##   Label Estimate       se         cv     lcl      ucl       df
## 1 Total 49.69687 2.936725 0.05909276 44.2033 55.87318 99.55689
```

```
# Fit alternative models
# Model 2. Uniform with cosine adjustments
nest.uf.cos <- ds(nests, key="unif", adjustment="cos",
                  convert.units=conversion.factor)
# Model 3. Hazard rate with hermite polynomial adjustments
nest.hr.herm <- ds(nests, key="hr", adjustment="herm",
                  convert.units=conversion.factor)
```

The goodness of fit for the basic model is shown below.

```
gof_ds(nest.hn, nc=8)
```




```
##
## Goodness of fit results for ddf object
##
## Distance sampling Cramer-von Mises test (unweighted)
```

```
## Test statistic = 0.0353634 p-value = 0.955416
```

A function useful for contrasting models is `summarize_ds_models`. A summary table of goodness of fit statistics for all models is created below. Note the code `pander::pander`: this is used because the output from `summarize_ds_models` is formatted for latex style editors and the `pander` function prints the output in a pretty way in a document.

```
# Summarise gof statistics
pander::pander(summarize_ds_models(nest.hn, nest.uf.cos, nest.hr.herm),
               caption="Model results for ducknest data set.")
```

Table 1: Model results for ducknest data set. (continued below)

Model	Key function	Formula
nest.hn	Half-normal	~1
nest.uf.cos	Uniform with cosine adjustment term of order 1	NA
nest.hr.herm	Hazard-rate	~1

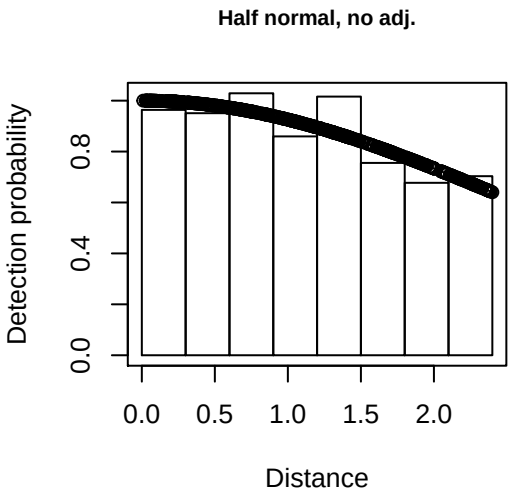
C-vM p-value	$\hat{P}_a$	se($\hat{P}_a$)	ΔAIC
0.9554	0.8693	0.03902	0
0.8208	0.8464	0.04407	0.3459
0.9806	0.8891	0.04958	1.66

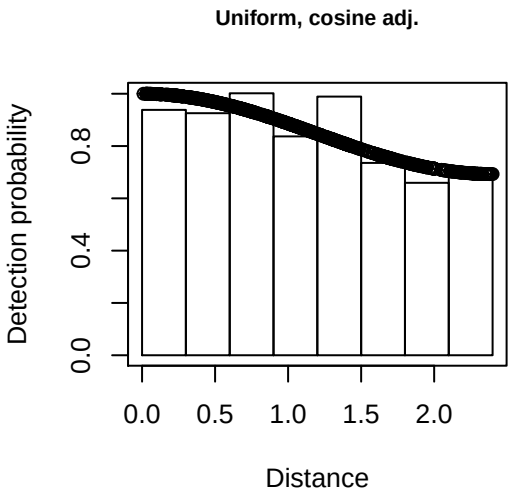
The density results from all models are summarized below.

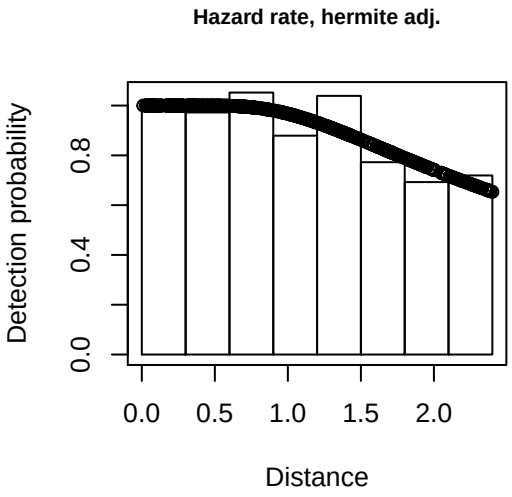
Model	DetectionFunction	AIC	Density	LowerCI	UpperCI
1	Half-normal, no adjustments	928.1	49.7	44.2	55.87
2	Uniform, cosine adjustments	928.5	51.04	44.92	58
3	Hazard rate, hermite adjustments	929.8	48.59	42.52	55.54

The detection function plots are shown below.

```
# Divide the plot window
par(mfrow=c(2,2))
# Plot detection functions
plot(nest.hn, nc=8, main="Half normal, no adj.")
plot(nest.uf.cos, nc=8, main="Uniform, cosine adj.")
plot(nest.hr.herm, nc=8, main="Hazard rate, hermite adj.")
```







The half-normal detection function with no adjustments has the smallest AIC which provides support for this model. The ΔAIC values for all three models is small. In general, you should get similar density estimates using different detection function models, provided those models fit the data well, as in this example.