Beginner Point-Transect Analysis in Rdistance

Michael Kleinsasser, Jason D. Carlisle and Trent L. McDonald

August 6, 2018

Introduction

This tutorial is a beginner's guide to doing point transect distance-sampling analysis using `Rdistance`. Topics covered include input data requirements, fitting a detection function, estimating abundance (or density), and selecting the best fit detection function using AICc. We use the internal datasets `thrasherDetectionData` and `thrasherSiteData` (point transect surveys of brown thrashers). This tutorial is current as of version 2.1.3 of `Rdistance`.

1: Install and load Rdistance

If you haven't already done so, install the latest version of `Rdistance`. In the R console, issue `install.packages("Rdistance")`. After the package is installed, it can be loaded into the current session as follows:

```
require(Rdistance)
```

2: Read in the input data

For this tutorial, we use two datasets collected by J. Carlisle on brown thrashers in central Wyoming that are included with `Rdistance`.

The first dataset, `thrasherDetectionData`, is a detection data.frame with one row for each detected object. Columns in the data frame are:

- `siteID` = Factor, the site (point) and transect surveyed. Levels are five character codes like 'TTXPP' where TT is transect number and PP is the point within the transect.
- `groupsize` = Numeric, the number of individuals within the detected group.
- `dist` = Numeric, the radial distance from the point to the detected group. Obtain access to the example dataset of thrasher detections and observed distances (`thrasherDetectionData`) using the following commands:

```
data("thrasherDetectionData")
head(thrasherDetectionData)
```

```
##   siteID groupsize dist
## 1  C1X01         1   11
## 2  C1X01         1  183
## 3  C1X02         1   58
## 4  C1X04         1   89
## 5  C1X05         1   83
## 6  C1X06         1   95
```

The second required dataset, `thrasherSiteData`, is a transect data.frame, with one row for each transect surveyed, and the following required columns:

- `siteID` = Factor, the site (point) and transect surveyed.

- ... = Any additional transect-level covariate columns (these will not be used in this tutorial).

Load the example dataset of thrasher transects (`thrasherSiteData`) using the following commands:

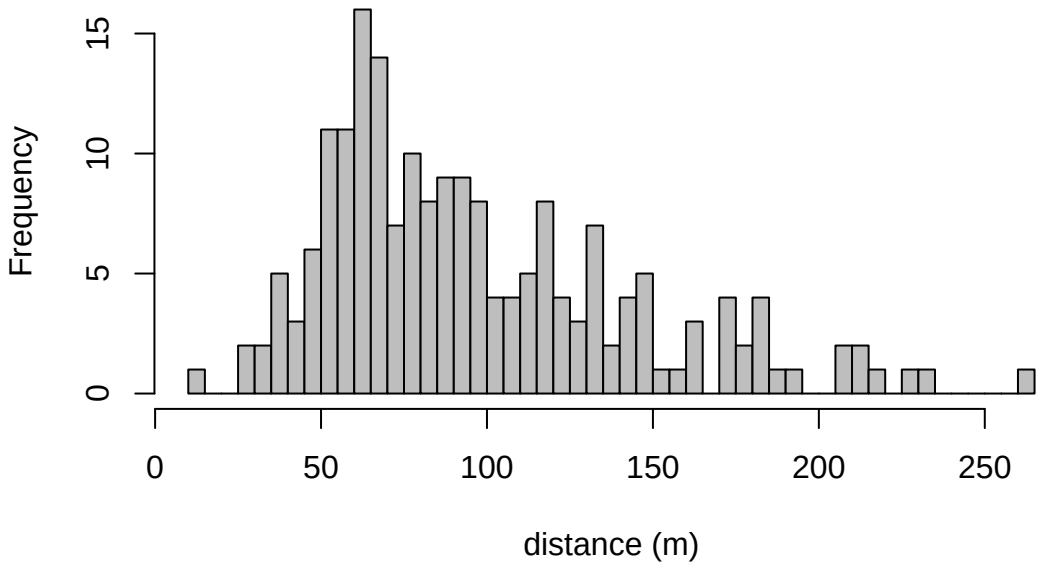
```
data("thrasherSiteData")
head(thrasherSiteData)
```

```
##   siteID observer bare herb shrub height
## 1  C1X01     obs5 45.8 19.5  18.7   23.7
## 2  C1X02     obs5 43.4 20.2  20.0   23.6
## 3  C1X03     obs5 44.1 18.8  19.4   23.7
## 4  C1X04     obs5 38.3 22.5  23.5   34.3
## 5  C1X05     obs5 41.5 20.5  20.6   26.8
## 6  C1X06     obs5 43.7 18.6  20.0   23.8
```

3: Fit a detection function

Once the data are imported, the first step is to fit a detection function. Before we do so, explore the distribution of the distances:

```
hist(thrasherDetectionData$dist, n=40, col="grey", main="", xlab="distance (m)")
```



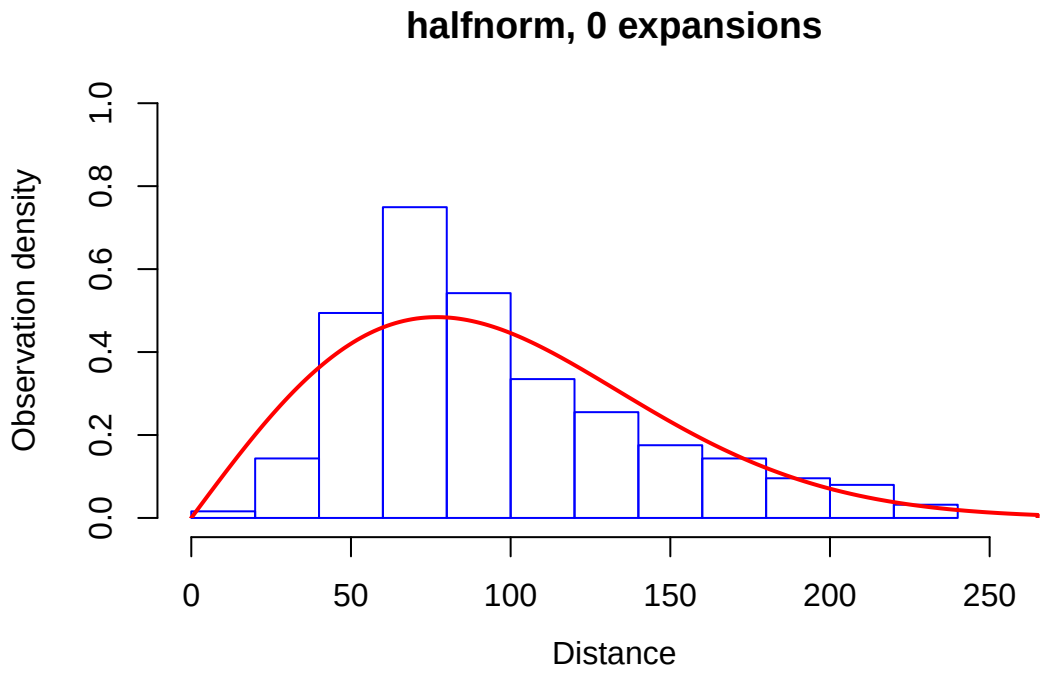

```
summary(thrasherDetectionData$dist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   63.00   86.00   97.16  123.00  265.00
```

Next, we fit a detection function using `dfuncEstim` to the radial distances collected from the point transects and plot it. We specify point transects using option `PointSurvey = TRUE` in the call to `dfuncEstim` and specify the the half-normal distance function using option `likelihood = "halfnorm"`. In section 5, we demonstrate an automated process to fit multiple detection functions and compare them using AICc.

```
dfunc <- dfuncEstim(formula = dist ~ 1,
                    detectionData = thrasherDetectionData,
                    pointSurvey = TRUE,
                    likelihood = "halfnorm")

plot(dfunc)
```




```
dfunc
```

```
## Call: dfuncEstim(formula = dist ~ 1, detectionData =
##    thrasherDetectionData, likelihood = "halfnorm", pointSurvey =
##    TRUE)
## Coefficients:
##        Estimate  SE        z         p(>|z|)
## Sigma  76.88767  2.906298  26.45553  3.151399e-154
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 265
## Effective detection radius (EDR): 108.5909
## Probability of detection: 0.1679172
## Scaling: g(0) = 1
## Log likelihood: 1004.254
## AICc: 2010.53
```

The effective detection radius (EDR) is the essential information from the detection function that will be used to estimate abundance in section 4. The EDR is calculated by integrating the detection function to compute area under the detection function. See the help documentation for `EDR` for details.

4: Estimate abundance given the detection function

Estimating abundance requires the additional information contained in the the thrasher site dataset, described in section 2, where each row represents one transect. Load the example dataset of surveyed thrasher transects from the package.

We estimate abundance (or density in this case) using `abundEstim`. If `area` = 1, density is given in the squared units of the distance measurements — in this case, thrashers per square meter. If we set `area` = 10000, we convert to thrashers per hectare (1 ha == 10,000 m^2). The equation used to calculate the abundance estimate is detailed in the help documentation for `abundEstim`.

Confidence intervals for abundance are calculated using a bias-corrected bootstrapping method (see `abundEstim`). Note that, as with all bootstrapping procedures, there may be slight differences in the confidence intervals between runs. Increasing the number of bootstrap iterations (`R` = 100 used here for brevity) may be necessary to stabilize CI estimates.

```
# Estimate Abundance - Density; fatalities per m2
fit <- abundEstim(dfunc          = dfunc,
                  detectionData = thrasherDetectionData,
                  siteData      = thrasherSiteData,
                  area           = 10000,       # density per hectare
                  R              = 100,
                  ci             = 0.95)
```

```
fit
```

```
## Call: dfuncEstim(formula = dist ~ 1, detectionData =
##    thrasherDetectionData, likelihood = "halfnorm", pointSurvey =
##    TRUE)
## Coefficients:
##        Estimate  SE        z         p(>|z|)
## Sigma  76.88767  2.906298  26.45553  3.151399e-154
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 265
## Effective detection radius (EDR): 108.5909
## Probability of detection: 0.1679172
## Scaling: g(0) = 1
## Log likelihood: 1004.254
## AICc: 2010.53
##
## Abundance estimate:  0.4408978 ;  95% CI=( 0.3520647 to 0.5176698 )
```

The abundance estimate can be extracted from the `fit` object.

```
fit$n.hat
```

```
## [1] 0.4408978
```

The confidence interval (in this case 95%) can be extracted from the `fit` object.

```
fit$ci
```

```
## 1.968631%  96.8533%
## 0.3520647 0.5176698
```

5: Use AICc to select a detection function and estimate abundance

Fitting several detection functions, choosing the best fitting, and estimating abundance (sections 3 and 4) can be automated using the function `autoDistSamp`. The function attempts to fit multiple detection functions, uses AICc (by default, but see help documentation for `autoDistSamp` under `criterion` for other options) to find the 'best' detection function, then proceeds to estimate abundance using the best fit detection function (the distance function with lowest AICc). By default, `autoDistSamp` tries a large subset of `Rdistance`'s built-in detection functions, but you can control exactly which detection functions are attempted (see help documentation for `autoDistSamp`). Specifying `plot=TRUE` produces a plot of each detection function as it is estimated. Specifying, `plot.bs=TRUE` plots the selected distance function each iteration of the bootstrap procedure. In this example, we fit the half-normal, hazard rate, exponential, and uniform likelihoods with no expansion terms, we do not plot all fitted functions (`plot=FALSE`), but we plot the best distance function fitted during each bootstrap iteration.

```
# Automated Fit - fit several models, choose the best model based on AIC
autoDS <- autoDistSamp(formula       = thrasherDetectionData$dist ~ 1,
                       detectionData = thrasherDetectionData,
                       siteData      = thrasherSiteData,
                       pointSurvey   = TRUE,
                       expansions    = c(0),
                       likelihoods   = c("halfnorm", "hazrate", "negexp", "uniform"),
                       plot          = FALSE,
                       area          = 10000,
                       R             = 100,
                       ci            = 0.95,
                       plot.bs       = FALSE)
```

```
autoDS
```

```
## Call: dfuncEstim(formula = formula, detectionData = detectionData,
##    siteData = siteData, likelihood = fit.table$like[1], pointSurvey
##    = pointSurvey, w.lo = w.lo, w.hi = w.hi, expansions =
##    fit.table$expansions[1], series = fit.table$series[1])
## Coefficients:
##        Estimate   SE         z         p(>|z|)
## Sigma  93.729520  5.8722732  15.96137  2.374641e-57
## Beta    4.199511  0.3971426  10.57431  3.920392e-26
##
## Convergence: Success
## Function: HAZRATE
## Strip: 0 to 265
## Effective detection radius (EDR): 118.6222
## Probability of detection: 0.2003736
## Scaling: g(0) = 1
## Log likelihood: 999.0199
## AICc: 2002.103
##
## Abundance estimate:  0.3694816 ;  95% CI=( 0.3468602 to 0.3949777 )
```

The detection function with the lowest AICc value (and thus selected as the 'best') is the hazard rate likelihood with 0 cosine expansion terms.

Conclusion

In sections 3 and 4, we fitted a half-normal detection function and used that function to estimate thrasher density. Our estimate was 0.44 thrashers per ha (95% CI = 0.35 to 0.52). In section 5, we used AICc to estimate a better fitting detection function and used it to estimate thrasher density. The thrasher density estimated by the better-fitting model was 0.37 thrashers per ha (95% CI = 0.35 to 0.39). (Note, CI estimates may vary slightly from these due to minor 'simulation slop' inherent in bootstrapping methods).