Beginner Line-Transect Analysis in Rdistance

Jason D. Carlisle, Abigail S. Hoffman, and Trent L. McDonald

May 23, 2018

Introduction

This tutorial is meant to be a beginner's guide to analyzing line-transect data using `Rdistance`. It is assumed that you have some familiarity with Program R, but not necessarily with distance-sampling analysis. This beginning tutorial focuses on input data requirements, fitting a detection function, and estimating abundance (or density). Here, we make use of the example datasets already contained within `Rdistance` (i.e., line transect surveys of sparrows), so you can complete this tutorial without having any data of your own. 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)
```

```
## Loading required package: Rdistance
```

```
## Rdistance (version 2.1.3)
```

2: Read in input data

`Rdistance` requires two input datasets. These can be prepared outside of `R` and read in as data.frames using, for example, `read.csv`. In the following sections, we make use of the sparrow example datasets already contained within `Rdistance`.

The first required dataset is a detection data.frame, with a row for each detection, and the following required columns, named as follows:

- `siteID` = Factor, the site or transect where the detection was made.
- `groupsize` = Numeric, the number of individuals within the detected group.
- `dist` = Numeric, the perpendicular distance (also known as off-transect distance) from the transect to the detected group.

If the observers recorded sighting distance and sighting angle instead of perpendicular distance (as is often common in line transect surveys), you can use the `perpDists` function (detailed in Section 3) to calculate the perpendicular distances based on the sighting distances and sighting angles.

The second required dataset is a transect data.frame, with a row for each transect surveyed, and the following required columns, named as follows:

- `siteID` = Factor, the site or transect surveyed.
- `length` = Numeric, the length of the transect. Use the same units as the detection distances.
- `shrubclass` = Factor, the height of sage shrubs (High or Low) where sparrow was detected
- `...` = Any additional transect-level covariate columns (these will be ignored).

3: Fit a detection function

After prepping the input data, the first step is to explore your data and fit a detection function. First, load the example dataset of sparrow detections and their distances from the package using `data(sparrowDetectionData)` and transect characteristices using `data(sparrowSiteData)`. Be sure that you have installed and loaded `Rdistance` prior to issuing the following commands:

```
data(sparrowDetectionData)
head(sparrowDetectionData)
```

```
##   siteID groupsize sightdist sightangle dist
## 1     A1         1        65         15 16.8
## 2     A1         1        70         10 12.2
## 3     A1         1        25         75 24.1
## 4     A1         1        40          5  3.5
## 5     A1         1        70         85 69.7
## 6     A1         1        10         90 10.0
```

```
data(sparrowSiteData)
head(sparrowSiteData)
```

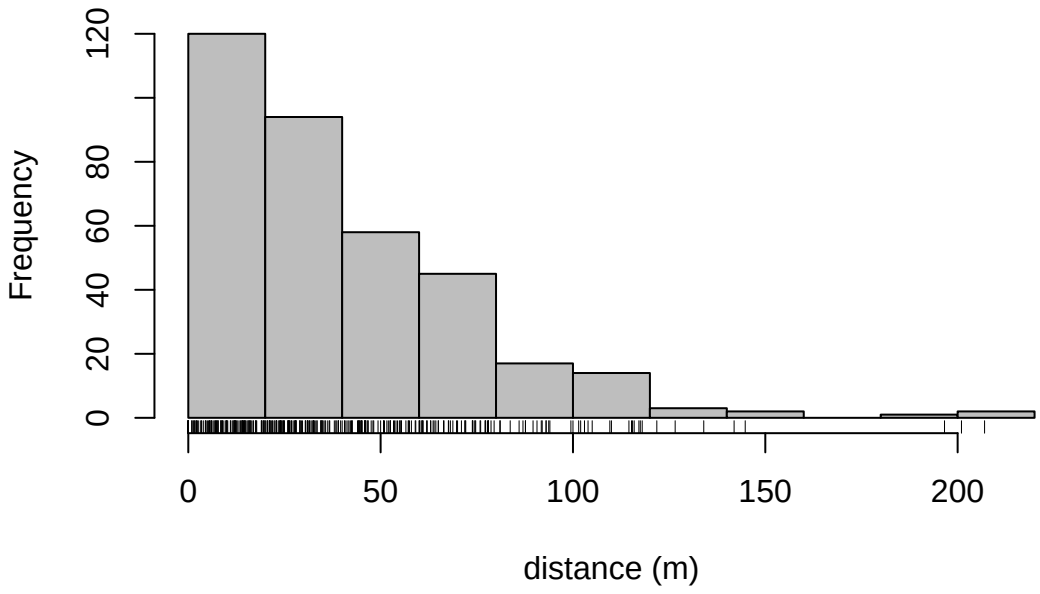
```
##   siteID length observer bare herb shrub height shrubclass
## 1     A1    500     obs4 36.7 15.9  20.1   26.4       High
## 2     A2    500     obs4 38.7 16.1  19.3   25.0       High
## 3     A3    500     obs5 37.7 18.8  19.8   27.0       High
## 4     A4    500     obs5 37.7 17.9  19.9   27.1       High
## 5     B1    500     obs3 58.5 17.6   5.2   19.6        Low
## 6     B2    500     obs3 56.6 18.1   5.2   19.0        Low
```

Line-transect distance-sampling analysis is done on perpendicular distances (i.e., the distance from each detected group to the transect, not to the observer). We have provided the perpendicular distances (named `dist`) in the example data, but observers originally recorded sighting distances and sighting angles. Here we use the `perpDists` function to (re)calculate the perpendicular distances (`dist`) and remove the `sightdist` and `sightangle` columns. See the help documentation for `perpDists` for details.

```
sparrowDetectionData$dist<- perpDists(sightDist="sightdist",
                    sightAngle="sightangle",
                    data = sparrowDetectionData)
sparrowDetectionData <- sparrowDetectionData[,
                    -which(names(sparrowDetectionData)
                    %in% c("sightdist", "sightangle"))]
```

Explore the distribution of distances.

```
hist(sparrowDetectionData$dist, col="grey", main="",
     xlab="distance (m)")
rug(sparrowDetectionData$dist,quiet = TRUE)
```



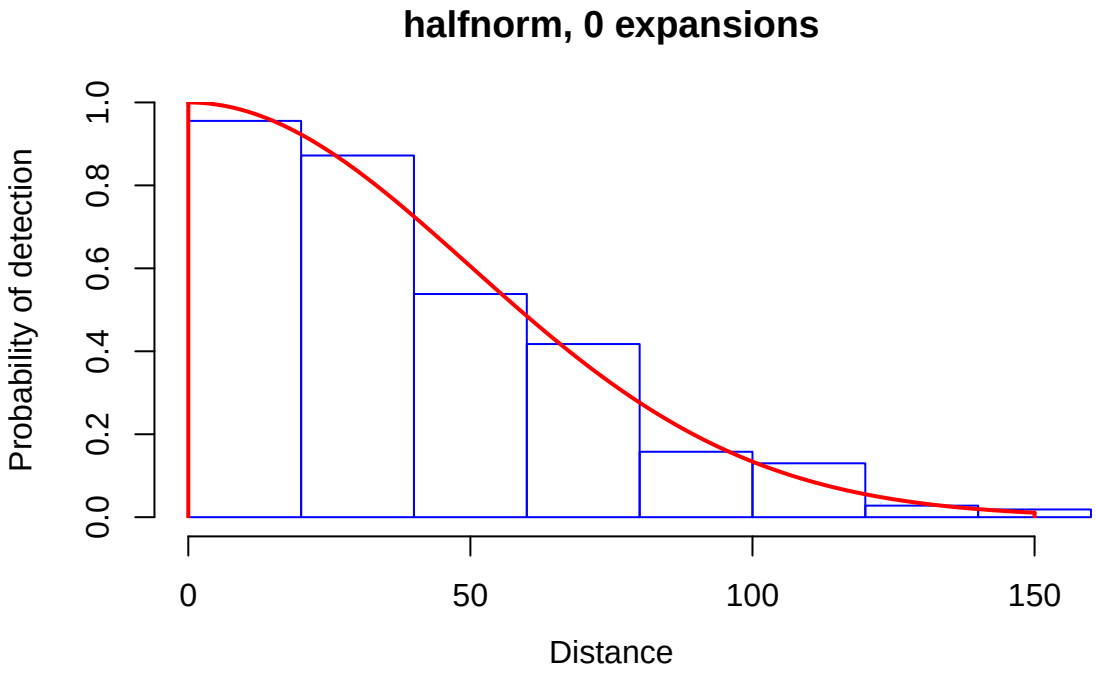

```
summary(sparrowDetectionData$dist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   14.19   30.76   39.64   57.33  207.00
```

Next, fit a detection function (plotted as a red line) using `dfuncEstim`. For now, we will proceed using the half-normal likelihood as the detection function, but in Section 5 of this tutorial, we demonstrate how to run an automated process that fits multiple detection functions and compares them using AICc. Note that distances greater than 150 m are quite sparse, so here we right-truncate the data, tossing out detections where `dist` > 150.

```
dfuncSparrow<- dfuncEstim(formula = dist~1,
                          detectionData = sparrowDetectionData,
                          siteData = sparrowSiteData,
                          likelihood = "halfnorm", w.hi = 150)

plot(dfuncSparrow)
```




```
dfuncSparrow
```

```
## Call: dfuncEstim(formula = dist ~ 1, detectionData =
##    sparrowDetectionData, siteData = sparrowSiteData, likelihood =
##    "halfnorm", w.hi = 150)
## Coefficients:
##        Estimate  SE        z         p(>|z|)
## Sigma  49.87246  2.014099  24.76167  2.321414e-135
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 150
## Effective strip width (ESW): 62.34128
## Probability of detection: 0.4156085
## Scaling: g(0) = 1
## Log likelihood: 1630.708
## AICc: 3263.428
```

The effective strip width (ESW) is the key information from the detection function that will be used to next estimate abundance (or density). The ESW is calculated by integrating under the detection function. A survey with imperfect detection and ESW equal to X effectively covers the same area as a study with perfect detection out to a distance of X. See the help documentation for `ESW` for details.

4: Estimate abundance given the detection function

Estimating abundance requires the additional information contained in the second required dataset, described earlier, where each row represents one transect.

Next, estimate abundance (or density in this case) using `abundEstim`. If `area`=1, then density is given in

the squared units of the distance measurements — in this case, sparrows per square meter. Instead, we set `area`=10000 in order to convert to sparrows 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`), and the detection function fit in each iteration of the bootstrap is plotted as a blue line (if `plot.bs=TRUE`). Note that, as with all bootstrapping procedures, there may be slight differences in the confidence intervals between runs due to simulation error.

Computing bootstrap confidence intervals for this vignette takes time (>5s), and the masters at CRAN have better things to do than wait for a package to build. In the code below, we supress bootstrap confidence intervals with `ci = NULL`, then replace confidence endpoints with values from a separate run with `R = 500` iterations.

```
fit <- abundEstim(dfuncSparrow,
                  detectionData=sparrowDetectionData,
                  siteData=sparrowSiteData,
                  area=10000,
                  ci=NULL)
# To save vignette build time, we insert values from
# a separate run with R=500
fit$ci <- c( lowCI=0.6473984, hiCI=1.0167007)
```

```
fit
```

```
## Call: dfuncEstim(formula = dist ~ 1, detectionData =
##    sparrowDetectionData, siteData = sparrowSiteData, likelihood =
##    "halfnorm", w.hi = 150)
## Coefficients:
##        Estimate  SE        z         p(>|z|)
## Sigma  49.87246  2.014099  24.76167  2.321414e-135
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 150
## Effective strip width (ESW): 62.34128
## Probability of detection: 0.4156085
## Scaling: g(0) = 1
## Log likelihood: 1630.708
## AICc: 3263.428
##
## Abundance estimate:  0.8265435 ;  % CI=( 0.6473984 to 1.016701 )
## CI based on 0 of 1 successful bootstrap iterations
```

Results of interest (such as the abundance estimate and confidence interval) can be extracted from the resulting object (here called `fit`).

```
fit$n.hat
```

```
## [1] 0.8265435
```

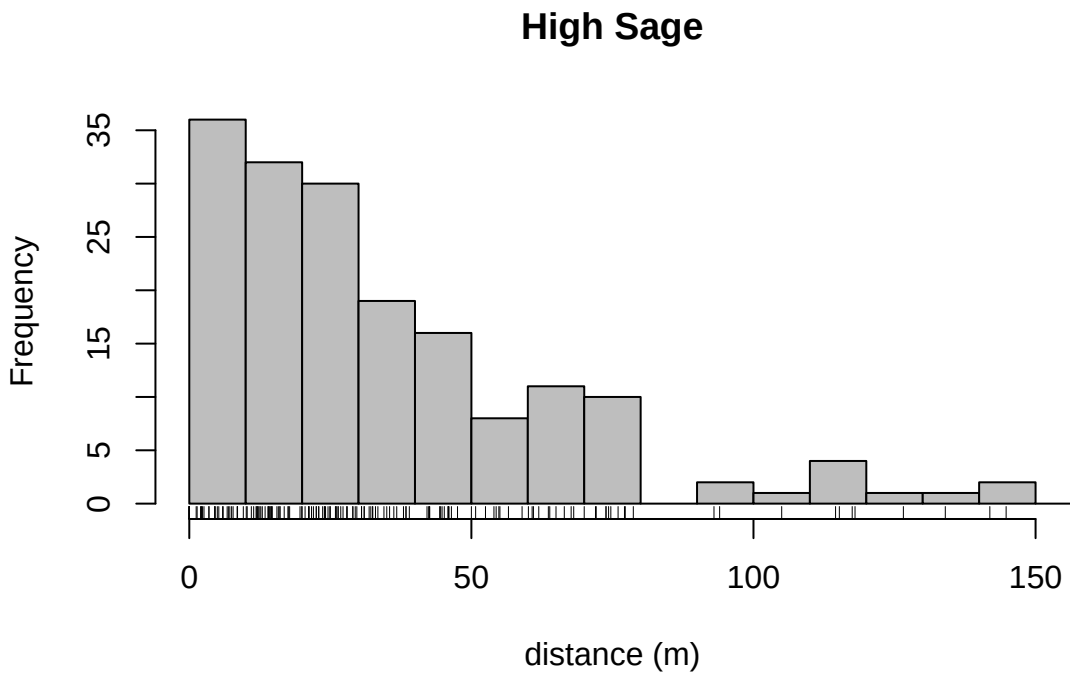
```
fit$ci
```

```
##     lowCI      hiCI
## 0.6473984 1.0167007
```

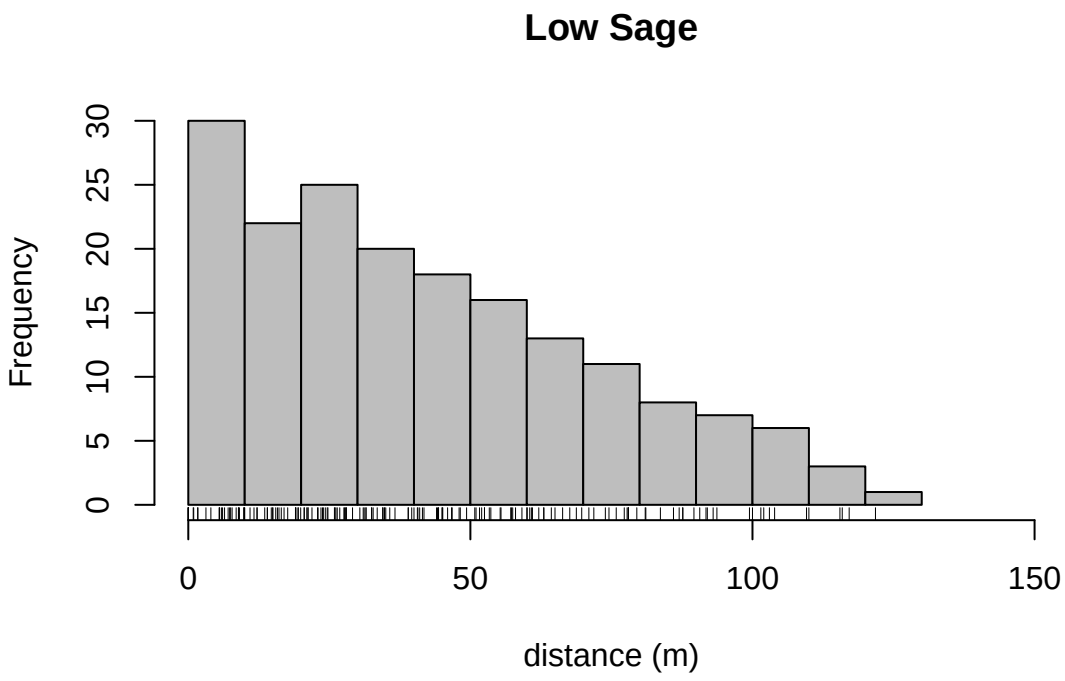
5: Estimate abundance using covariates

For some data, the probability of detection may be dependent upon covariates due to site characteristics such as vegetation type or object characteristics such as animal size. Explore the distribution of distances for sparrows detected in different sagebrush shrub classes.

```
hist(sparrowDetectionData$dist[sparrowSiteData$shrubclass ==
     "High"], col="grey", main="High Sage",
     xlab="distance (m)", breaks = 15, xlim = c(0,150))
rug(sparrowDetectionData$dist[sparrowSiteData$shrubclass ==
    "High"],quiet = TRUE)
```



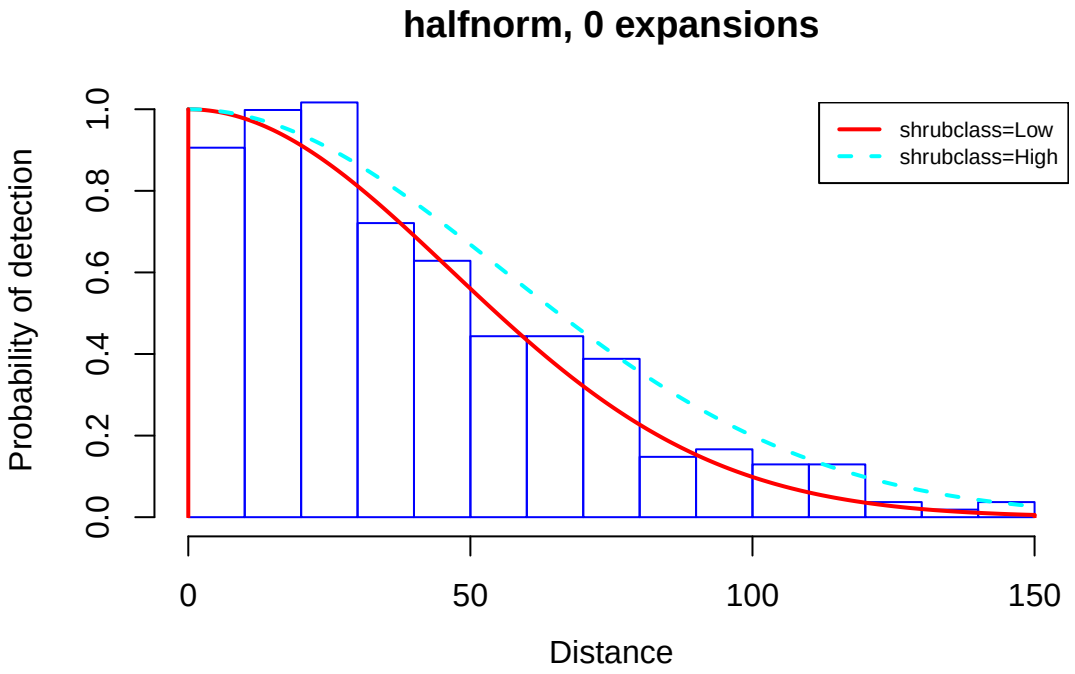

```
hist(sparrowDetectionData$dist[sparrowSiteData$shrubclass ==
    "Low"], col="grey", main="Low Sage",  xlab="distance (m)",
     breaks = 14, xlim = c(0,150))
rug(sparrowDetectionData$dist[sparrowSiteData$shrubclass ==
    "Low"],quiet = TRUE)
```



We can fit detection functions to these different site or object classes to better estimate abundance. We can specifiy different detection functions for these classes by using the formula argument in `dfuncEstim`. For the sparrow data, we fit detection functions to observations in high and low sagebrush shrub classes. Like the previous example, to save vignette build time, we supress bootstrap sampling then substitute values from a separate run with `R=500`.

```
dfuncShrubClass<- dfuncEstim(formula = dist~shrubclass,
                          detectionData = sparrowDetectionData,
                          siteData = sparrowSiteData,
                          likelihood = "halfnorm", w.hi = 150)
```

```
plot(dfuncShrubClass, newdata = data.frame(shrubclass=
                levels(sparrowSiteData$shrubclass)), nbins = 15)
```




```
covarFit <- abundEstim(dfuncShrubClass,
                       detectionData=sparrowDetectionData,
                       siteData=sparrowSiteData,
                       area=10000,
                       ci=NULL)
# To save vignette build time, insert values from
# a separate run with R=500
covarFit$ci <- c("2.939477%" = 0.6552051, "97.88301%"= 1.0411813)
```

```
covarFit
```

```
## Call: dfuncEstim(formula = dist ~ shrubclass, detectionData =
##    sparrowDetectionData, siteData = sparrowSiteData, likelihood =
##    "halfnorm", w.hi = 150)
## Coefficients:
##                 Estimate    SE          z          p(>|z|)
## (Intercept)      4.0196980  0.07090136  56.694228  0.00000000
## shrubclassHigh  -0.1813069  0.08649500  -2.096154  0.03606852
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 150
## Average effective strip width (ESW): 62.28349
## Average probability of detection: 0.4152232
## Scaling: g(0) = 1
## Log likelihood: 1628.379
## AICc: 3260.792
##
## Abundance estimate:  0.8350688 ;  % CI=( 0.6552051 to 1.041181 )
## CI based on 0 of 1 successful bootstrap iterations
```

In this example including the covariate of shrub class changes our estimate from 0.83 sparrows per ha (95% CI = 0.65 - 1.02) to 0.84 sparrows per ha (95% CI = 0.66 - 1.04). In this case, the density estimate did not change much when we included `shrubClass`, but this will not always be the case. Covariates can have a large influence on abundance estimates.

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

Alternatively, steps 3 (fitting a detection function) and 4 (estimating abundance) can be automated using the function `autoDistSamp`. This function attempts to fit multiple detection functions, uses AICc (by default, but see help documentation for `AIC.dfunc` for other options) to find the 'best' detection function, then proceeds to estimate abundance using that detection function. 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` would return a plot of each detection function. In this example, we fit the three detection function forms combined with 0 or 1 expansion terms, and we don't plot them (`plot=FALSE`). If `autoDistSamp` is called from another function, it can be convenient to suppress all output by setting `plot.bs=FALSE`, `plot=FALSE`, and `showProgress=FALSE`.

```
auto <- autoDistSamp(formula        = dist~1,
                     detectionData  = sparrowDetectionData,
                     siteData       = sparrowSiteData,
                     area           = 10000,
                     likelihoods    = c("halfnorm","hazrate","uniform"),
                     expansions     = 0:1,
                     series         = "cosine",
                     ci             = 0.95,
                     R              = 500,
                     plot           = FALSE,
                     plot.bs        = FALSE,
                     w.hi           = 150)
```

```
## Likelihood   Series  Expans  Converged?  Scale?  AICc
## halfnorm cosine  0   Yes     Ok  3263.443
## halfnorm cosine  1   Yes     Ok  3261.5853
## hazrate      cosine  0   Yes     Ok  3267.6249
## hazrate      cosine  1   Yes     Ok  3263.3098
## uniform      cosine  0   Yes     Ok  3260.7321
## uniform      cosine  1   Yes     Ok  3262.4814
## Computing bootstrap confidence interval on N...
## |======================================================================| 100%
## 101 of 500 iterations did not converge.
##
## ------------ Abundance Estimate Based on Top-Ranked Detection Function ------------
## 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|)
## Threshold  27.7397495  19.537302641  1.419835  1.556557e-01
## Knee        0.0340005   0.005304744  6.409451  1.460443e-10
##
## Convergence: Success
## Function: UNIFORM
```

```
## Strip: 0 to 150
## Effective strip width (ESW): 51.34584
## Probability of detection: 0.3423056
## Scaling: g(0) = 1
## Log likelihood: 1628.349
## AICc: 3260.732
##
## Abundance estimate:  1.003543 ;  95% CI=( 0.7823177 to 1.366216 )
## CI based on 399 of 500 successful bootstrap iterations
```

You can see that the detection function with the lowest AICc value (and thus selected as the 'best') is the uniform likelihood. It is worth noting that approximately 20% of the bootstrap iterations did not converge, indicating that the uniform detection function is unstable. It might be best in this situation to exclude the uniform from the list of potential distance function forms.

We can also estimate abundances with covariates using the function `autoDistSamp`. However, automatic selection of covariates does not (yet) occur and none of the likelihoods accept expansions when covariates are present. In this example, we include a covariate in the distance function (`shrubclass`) and fit three different forms (`likelihoods = c("halfnorm", "hazrate", "negexp")`) without expansions (`expansions = 0`).

```
autoCov <- autoDistSamp(formula     = dist~shrubclass,
                        detectionData  = sparrowDetectionData,
                        siteData       = sparrowSiteData,
                        likelihoods    = c("halfnorm", "hazrate", "negexp"),
                        expansions     = 0,
                        area           = 10000,
                        ci             = 0.95,
                        R              = 500,
                        plot           = FALSE,
                        plot.bs        = FALSE,
                        w.hi           = 150)
```

```
## Likelihood Series  Expans  Converged?  Scale?  AICc
## halfnorm   cosine  0       Yes         Ok      3260.7917
## hazrate    cosine  0       Yes         Ok      3264.531
## negexp     cosine  0       Yes         Ok      3260.8539
## Computing bootstrap confidence interval on N...
## |======================================================================| 100%
##
## ------------ Abundance Estimate Based on Top-Ranked Detection Function ------------
## 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|)
## (Intercept)      4.0196980  0.07090136  56.694228  0.00000000
## shrubclassHigh  -0.1813069  0.08649500  -2.096154  0.03606852
##
## Convergence: Success
## Function: HALFNORM
## Strip: 0 to 150
## Average effective strip width (ESW): 62.28349
## Average probability of detection: 0.4152232
## Scaling: g(0) = 1
## Log likelihood: 1628.379
```

```
## AICc: 3260.792
##
## Abundance estimate:  0.8350688 ;  95% CI=( 0.6484067 to 1.036907 )
```

Conclusion

Note that the detection function that you select can have a large influence on the resulting abundance estimate. In sections 3 and 4, we fit a half-normal detection function and used that function to estimate sparrow density. Our estimate was 0.83 sparrows per ha (95% CI = 0.65 - 1.02). In section 5, we included `shrubClass` in the model and found that the average density estimate changed little. This will not always be the case as inclusion of covariates can have a substantial effect on estimates. In section 6, we used AICc to determine the best-fitting detection function amoung several we chose to fit and used that function to estimate sparrow density again. Initially, AICc determined that the uniform likelihood fit best and estimated density at 1 sparrows per ha (95% CI = 0.78 - 1.37). But, the high number of non-convergent bootstap iterations lead to some suspicion of the initial model. We added `shrubClass` into the model, dropped the uniform likelihood from consideration and AICc determined that the half normal detection function fit better than the hazard rate and negative exponential, but not by much. This latter model, also fitted in section 5, estimated density at 0.84 sparrows per ha (95% CI = 0.65 - 1.04). (Note, recall that your estimates may vary slightly from these due to variation inherent in bootstrapping methods). We conclude that the half normal distance function with `shrubClass` is a reasonable model for these data and that different functional forms for the distance function can have a substantial impact on abundance estimates. The `autoDistSamp` function can help select a detection function based on AICc, but ultimately the analyst will need to make the final distance function choice based on plots and other diagnostics.