Review of field sampling techniques

Point count data analysis workshop 2025

Péter Sólymos

2025-11-12

Table of contents

Preamble

```
suppressPackageStartupMessages({
    library(dplyr)
    library(ggplot2)
    library(unmarked)
    library(mefa4)
    library(detect)
})
```

Will use Decid * ConifWet

Species data

Data from Mahon et al. 2016 and Mahon et al. 2019.

Mahon, C. L., Holloway, G., Solymos, P., Cumming, S. G., Bayne, E. M., Schmiegelow, F. K. A., Song, S. J., 2016. Community structure and niche characteristics of upland and lowland western boreal birds at multiple spatial scales Forest Ecology and Management 361:99-116. https://doi.org/10.1016/j.foreco.2015.11.007

Mahon, C. L., Holloway, G., Bayne, E. M., Toms, J. D., 2019. Additive and interactive cumulative effects on boreal landbirds: winners and losers in a multi-stressor landscape Ecological Applications 29:e01895. https://doi.org/10.1002/eap.1895

```
d <- detect::josm$counts |>
    mutate(Behav = DetectType1, Count = 1) |>
    select(SiteID, SpeciesID, Count, Behav, Dur, Dis) |>
    arrange(SiteID, SpeciesID, Dur, Dis)
rownames(d) <- NULL
```

Point count data is usually organized in long format with the following information:

- the ID of the site or survey visit (we use `SiteID` because only 1 station and visit was done at each site)
- the species code (4 letters)
- the count (in this case each row is a distinct individual)
- the behavior code (C=call, S=song, V=visual)
- the time interval when the individual was first detected
- the distance interval in which the individual was first detected

```
summary(d)
```

```
     SiteID         SpeciesID          Count    Behav          Dur
 CO10712:  206   TEWA    : 5635   Min.   :1   C: 9180   0-3min :33648
 CO12627:  107   YRWA    : 4134   1st Qu.:1   S:41808   3-5min : 7220
 CO12724:   96   OVEN    : 4048   Median :1   V: 1384   5-10min:11504
 CO12780:   87   WTSP    : 3753   Mean   :1
 FT06718:   87   SWTH    : 3402   3rd Qu.:1
 CO12613:   85   CHSP    : 2096   Max.   :1
 (Other):51704   (Other):29304
      Dis
 0-50m  :23531
 50-100m:22230
 100+m  : 6611
```

```
str(d)
```

```
'data.frame':   52372 obs. of  6 variables:
 $ SiteID   : Factor w/ 4569 levels "CL10102","CL10106",..: 1 1 1 1 1 1 1 2 2 ...
 $ SpeciesID: Factor w/ 149 levels "ALFL","AMBI",..: 43 46 95 95 95 107 107 140 21 38 ...
 $ Count    : num  1 1 1 1 1 1 1 1 1 ...
 $ Behav    : Factor w/ 3 levels "C","S","V": 1 1 2 2 2 2 2 2 2 ...
 $ Dur      : Factor w/ 3 levels "0-3min","3-5min",..: 3 3 1 1 1 1 1 3 1 ...
 $ Dis      : Factor w/ 3 levels "0-50m","50-100m",..: 3 3 1 2 2 1 1 2 1 1 ...
```

```
d |> filter(SiteID == "CL10118", SpeciesID == "OVEN")
```

```
   SiteID SpeciesID Count Behav     Dur     Dis
1 CL10118      OVEN     1     S  0-3min 50-100m
2 CL10118      OVEN     1     S  0-3min 50-100m
3 CL10118      OVEN     1     S  0-3min   100+m
4 CL10118      OVEN     1     S 5-10min   0-50m
```

We can work with a single species or multiple species. Notice that `SiteID`, `SpeciesID`, etc. are coded as factors. This allows us to keep all the `SiteID` even after subsetting. If we keep these columns as plain character, we have to remember that there are no null records, i.e. sites where the species was not detected. The long format only contains information about detections.

If for some reason, no individuals of any species were detected at a site, we would add a `"NONE"` placeholder for the `SpeciesID`.

Point count duration

Each point-count in this data set has a total duration of 10 minutes, and 3 time intervals: 0–3, 3–5, and 5–10 minutes.

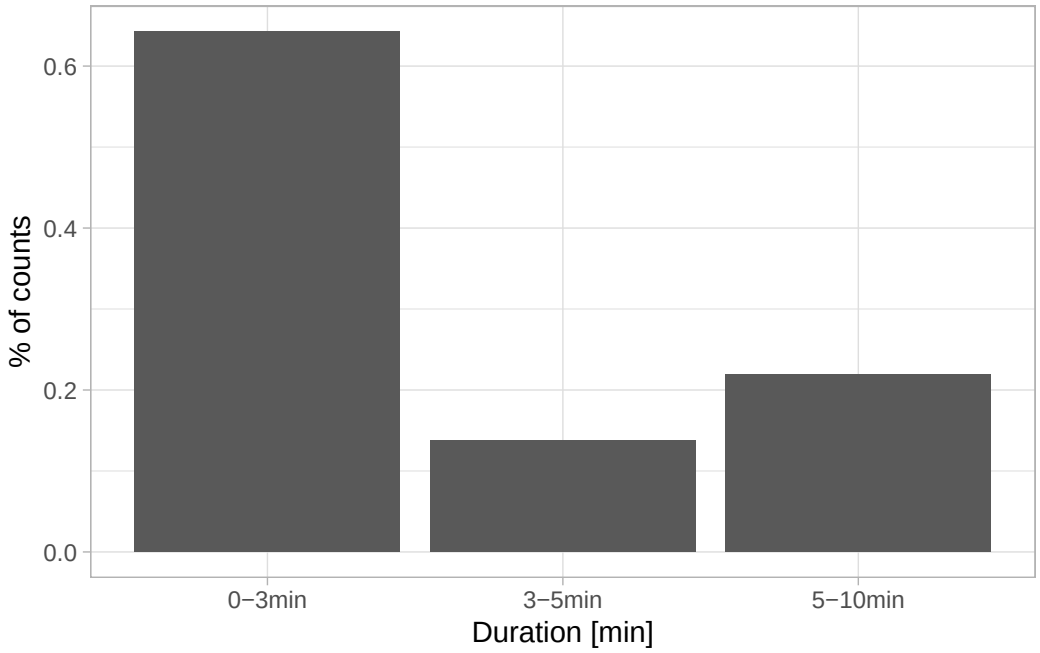
We used the `table()` function to tabulate the frequency of counts (rows) by time interval.

```
100 * table(d$Dur) / nrow(d)
```

```
  0-3min   3-5min  5-10min
64.24807 13.78599 21.96594
```

This is similar to `tally()`:

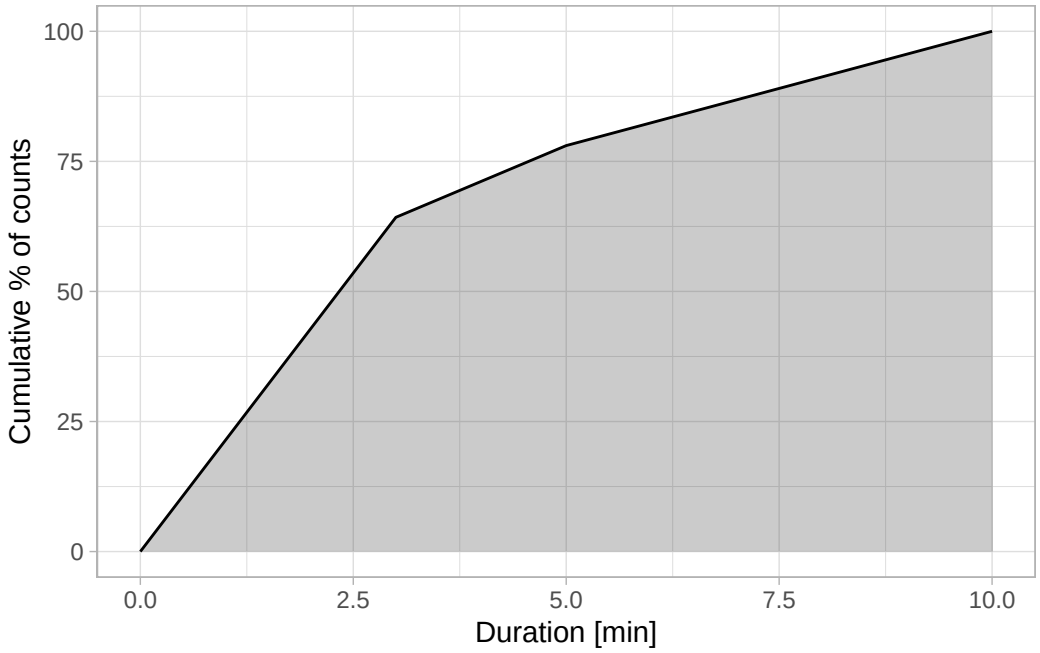
```
d |>
    group_by(Dur) |>
    tally() |>
    mutate(
        Perc = n / sum(n)
    ) |>
    ggplot(aes(x = Dur, y = Perc)) +
    geom_bar(stat = "identity") +
    xlab("Duration [min]") +
    ylab("% of counts") +
    theme_light()
```



We see most of the counts in the first 3 minutes, least of the counts in the next 2 minutes, and the rest th the second half of the total 10 minutes duration. These counts represent new individuals.

We can also plot this as a cumulative percentage:

```r
data.frame(
    Dur = c(0, 3, 5, 10),
    Cumul = c(0, 100 * cumsum(table(Dur = d$Dur) / nrow(d)))
) |>
    ggplot(aes(x = Dur, y = Cumul)) +
    # geom_bar(stat = "identity") +
    geom_area(alpha = 0.25) +
    geom_line() +
    xlab("Duration [min]") +
    ylab("Cumulative % of counts") +
    theme_light()
```



Point count radius

The data set includes 3 distance bands:

- 0–50 m
- 50–100 m
- 100 m

This is a so called unlimited distance count, that is quite common.

When the individuals are counted to an unknown distance (infinity), It is hard to define the survey area. We'll show some ways of estimating it later.

Sometimes a 400 m limit is assumed as the maximum distance from which cues can be heard, which is why BBS stations are 800 m (half mile) apart.

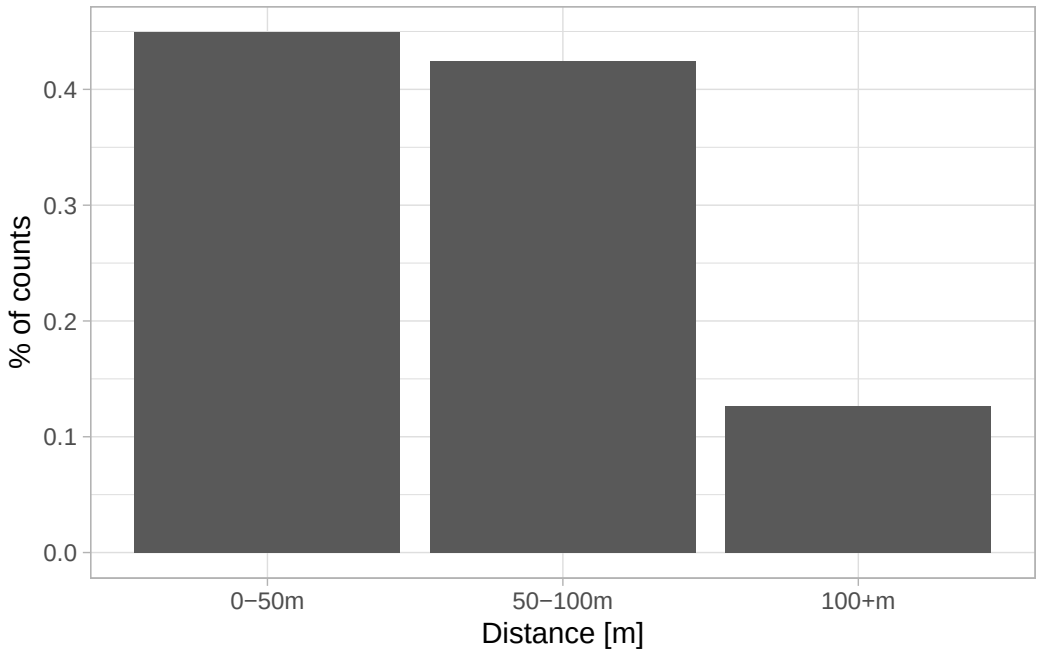
For truncated radius counts, the maximum distance within which the individuals are detected, the area is known. E.g. if we use the 0–50 m circle, the area is $A = (50m)^2 * \pi$. For the first 2 distance bands combined, $A = (100m)^2 * \pi$.

```
100 * table(d$Dis) / nrow(d)
```

```
   0-50m  50-100m    100+m
44.93050 42.44635 12.62316
```

Most of the counts were in the 0–50 m bin, but almost equal frequency was counted in the 50–100 m bin, which is only 3 times larger area. The smallest portion is outside of the 100 m distance.

```
d |>
    group_by(Dis) |>
    tally() |>
    mutate(Perc = n / sum(n)) |>
    ggplot(aes(x = Dis, y = Perc)) +
    geom_bar(stat = "identity") +
    xlab("Distance [m]") +
    ylab("% of counts") +
    theme_light()
```



We can also tabulate both the distance and duration bins:

```
100 * round(table(Dis = d$Dis, Dur = d$Dur) / nrow(d), 3) |>
    addmargins()
```

```
          Dur
Dis        0-3min 3-5min 5-10min   Sum
  0-50m      30.7    5.3     8.9  44.9
  50-100m    27.0    6.1     9.4  42.5
  100+m       6.5    2.4     3.7  12.6
  Sum        64.2   13.8    22.0 100.0
```

Working with a single species

Working with subsets of the data frame using Tidyverse functions can be misleading because non-detections are not explicitly stored.

```
y1 <- d |>
    filter(SpeciesID == "OVEN") |>
    group_by(SiteID) |>
    summarize(y = n())
dim(y1)
```

```
[1] 2077    2
```

Better to use functions that keep site ID's for non-detections. For example the `Xtab()` function from the mefa4 package.

```
y2 <- mefa4::Xtab(Count ~ SiteID + SpeciesID, d)[, "OVEN", drop = FALSE]
dim(y2)
```

```
[1] 4569    1
```

Let's see the total counts by distance and duration intervals:

```
y3 <- mefa4::Xtab(Count ~ Dis + Dur, d, subset = d$SpeciesID == "OVEN")
100 * y3 / sum(y3)
```

```
3 x 3 sparse Matrix of class "dgCMatrix"
          0-3min   3-5min  5-10min
0-50m   27.47036 2.692688 3.162055
50-100m 42.09486 6.052372 7.411067
100+m    6.59585 1.926877 2.593874
```

It is also possible to tabulate counts in a 3-way fashion, e.g. calculate the distance x duration combinations for each site. The first 2 dimensions will be treated as rows and columns of matrices. The 3rd dimension is used is used to form a named list.

```
y4 <- mefa4::Xtab(Count ~ Dis + Dur + SiteID, d, subset = d$SpeciesID == "OVEN")
y4[1:2]
```

```
$CL10102
3 x 3 sparse Matrix of class "dgCMatrix"
        0-3min 3-5min 5-10min
0-50m        1      .       .
50-100m      2      .       .
100+m        .      .       .

$CL10106
3 x 3 sparse Matrix of class "dgCMatrix"
        0-3min 3-5min 5-10min
0-50m        .      .       .
50-100m      .      .       .
100+m        .      .       .
```

Slightly more useful (and faster) to use site IDs as the 1st dimension, though:

```
y5 <- mefa4::Xtab(Count ~ SiteID + Dis + Dur, d, subset = d$SpeciesID == "OVEN")
lapply(y5, head, 2)
```

```
$`0-3min`
2 x 3 sparse Matrix of class "dgCMatrix"
        0-50m 50-100m 100+m
CL10102     1       2     .
CL10106     .       .     .

$`3-5min`
2 x 3 sparse Matrix of class "dgCMatrix"
        0-50m 50-100m 100+m
CL10102     .       .     .
CL10106     .       .     .

$`5-10min`
2 x 3 sparse Matrix of class "dgCMatrix"
        0-50m 50-100m 100+m
CL10102     .       .     .
CL10106     .       .     .
```

We can calculate different sums:

```
# 3min 50m
table(y5[["0-3min"]][, "0-50m"])
```

```
   0    1    2    3    4
3654  737  161   15    2
```

```
# 5min 100m
table(rowSums(y5[["0-3min"]][, 1:2]) + rowSums(y5[["3-5min"]][, 1:2]))
```

```
   0    1    2    3    4    5    6
2784  846  603  246   73   14    3
```

```
# 10min Inf
table(rowSums(y5[[1]]) + rowSums(y5[[2]]) + rowSums(y5[[3]]))
```

```
   0    1    2    3    4    5    6
2492  881  654  365  134   30   13
```

Working with multiple species

To get usable species data, we need to pivot the data. The crosstabulation uses sparse matrices (0's are not stored), which leads to smaller memory footprint and increased speed. This approach exploits the fact that fill rate (proportion of non-zero values) of bird count data is pretty low (~5% in this case).

```
ytot <- mefa4::Xtab(Count ~ SiteID + SpeciesID, d)

head(ytot[, 1:10])
```

```
6 x 10 sparse Matrix of class "dgCMatrix"

  [[ suppressing 10 column names 'ALFL', 'AMBI', 'AMCO' ... ]]


CL10102 . . . . . . . . . .
CL10106 . . . . . . . . . .
CL10108 2 . . . . . . . . .
CL10109 . . . 2 . . . . . .
CL10111 . . . . . . 1 . . .
CL10112 . . . . . . . . . .
```

```
mean(ytot > 0)
```

```
[1] 0.05014388
```

A sparse matrix can be converted to ordinary matrix

```
head(as.matrix(ytot)[, 1:10])
```

```
         ALFL AMBI AMCO AMCR AMGO AMKE AMRE AMRO AMWI ATTW
CL10102    0    0    0    0    0    0    0    0    0    0
CL10106    0    0    0    0    0    0    0    0    0    0
CL10108    2    0    0    0    0    0    0    0    0    0
CL10109    0    0    0    2    0    0    0    0    0    0
CL10111    0    0    0    0    0    0    1    0    0    0
CL10112    0    0    0    0    0    0    0    0    0    0
```

Check if counts are as expected:

```
max(ytot) # this is interesting
```

```
[1] 200
```

```
rev(sort(apply(as.matrix(ytot), 2, max)))[1:10] # it is CAGO
```

```
CAGO BLTE RECR PISI WWCR CEDW BLBW DCCO GRAJ AMPI
 200   70   51   50   40   23   20   19   16   12
```

```
## flyover (FO) flock (FL) beyond 100m distance
detect::josm$counts |>
    filter(
        SiteID == rownames(ytot)[which(ytot[, "CAGO"] == 200)],
        SpeciesID == "CAGO"
    ) |>
    head(2)
```

```
                      ObservationID  SiteID StationID TimeInterval Direction
CO10712-130603-008 CO10712-130603-008 CO10712 CO10712-1            1         2
CO10712-130603-009 CO10712-130603-009 CO10712 CO10712-1            1         2
                   Distance DetectType1 DetectType2 DetectType3 Sex Age
CO10712-130603-008        3           C        <NA>        <NA>   U   A
CO10712-130603-009        3           C        <NA>        <NA>   U   A
                   Activity1 Activity2 Activity3 ActivityNote    Dur   Dis
CO10712-130603-008        FO        FL      <NA>         <NA> 0-3min 100+m
CO10712-130603-009        FO        FL      <NA>         <NA> 0-3min 100+m
                   SpeciesID
CO10712-130603-008      CAGO
CO10712-130603-009      CAGO
```

The nice thing about this cross tabulation is that we can filter the records without changing the structure (rows, columns) of the table:

```
ytot_seen <- Xtab(Count ~ SiteID + SpeciesID, d, subset = d$Behav == "V")
ytot_heard <- Xtab(Count ~ SiteID + SpeciesID, d, subset = d$Behav != "V")
```

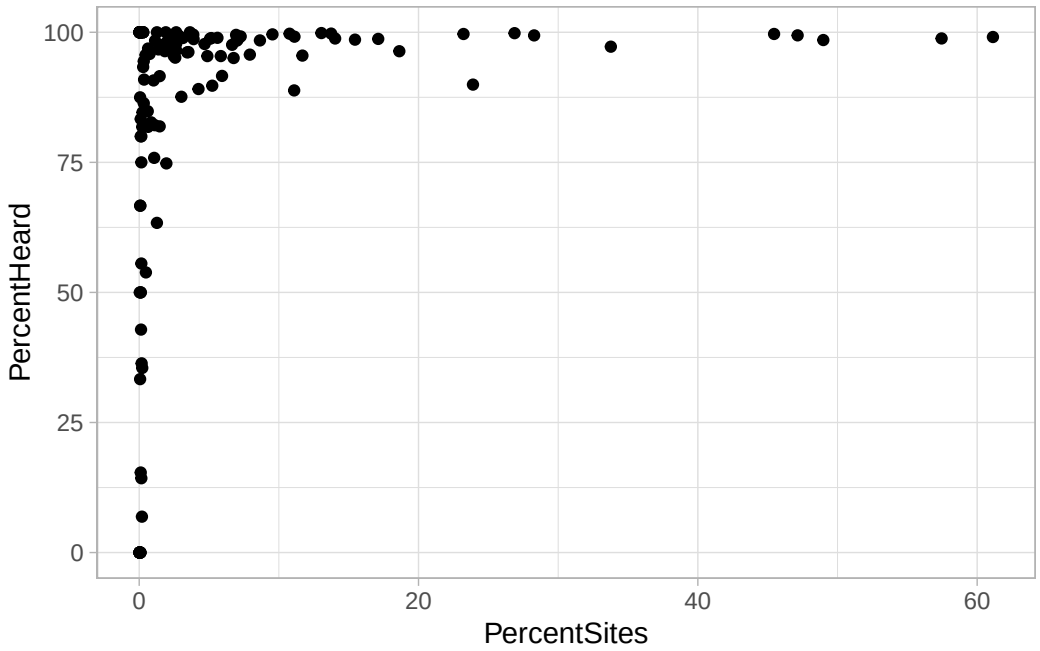
We see that the species that are detected only via visual cues are rare, mostly raptors and waterfowls:

```
ph <- data.frame(
    PercentSites = 100 * colSums(ytot > 0) / nrow(ytot),
    PercentHeard = 100 * colSums(ytot_heard) / colSums(ytot),
    SpeciesID = colnames(ytot)
) |>
    arrange(PercentHeard) |>
    left_join(detect::josm$species, by = join_by(SpeciesID))

head(ph)
```

```
  PercentSites PercentHeard SpeciesID              SpeciesName
1   0.08754651            0      BUFF                Bufflehead
2   0.08754651            0      BWTE         Blue-winged Teal
3   0.04377325            0      COGO         Common Goldeneye
4   0.02188663            0      COHA            Cooper's Hawk
5   0.04377325            0      DCCO Double-crested Cormorant
6   0.08754651            0      GWTE        Green-winged Teal
         ScientificName
1     Bucephala albeola
2          Anas discors
3    Bucephala clangula
4    Accipiter cooperii
5 Phalacrocorax auritus
6           Anas crecca
```

```
ph |>
    ggplot(aes(x = PercentSites, y = PercentHeard)) +
    geom_point() +
    theme_light()
```



The columns in these matrices contain the species counts, and will serve as the response variable (also called dependent variable) in the models that we fit next.

Next

Overview of regression techniques