Distance sampling in R

Next week: lab mid-term!!!

Lab mid-term

- Don’t miss it
- During regular lab time
- First hour: lab exercises
- Second hour: lab exam – work on a set of R tasks similar to homework assignments
- At the end of the hour, knit report and submit on Canvas
- To study, review lab and homework code
- Remember: you will have access to the internet to use help pages, look up R cheat sheets and so forth
- Remember basic object types (variables, vectors, matrices, data frames, lists) and variable types (numeric, character, factor, logical), as well as basic commands (arithmetic and logical operators, reading in data, creating objects, etc)

Distance sampling

Basic principles

- When detection declines with increasing distance from a transect of width *w*, detection is not perfect across all distances in *w* → *n*/*Lw* is negatively biased estimate of density

Density estimation, distance sampling:

- $$\widehat{D} = \frac{n}{2L\widehat{\mu}}$$
- *n*= number of detected individuals
- *L*=transect length
- $\hat{\mu}$= effective half-width (estimate)

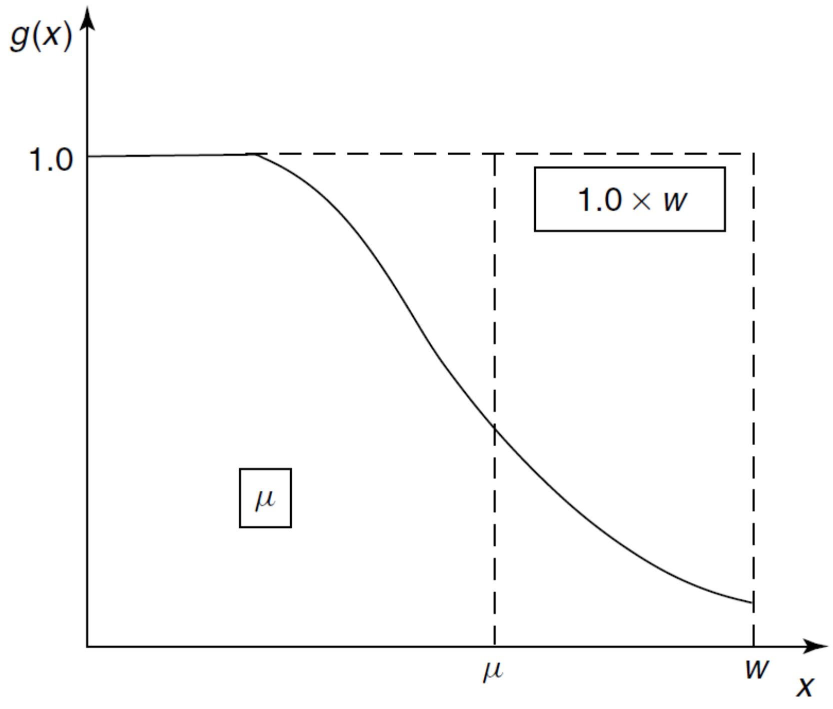
Distance sampling

Detection function

- Describes decline in detection probability with distance from transect
- Detection assumed to be perfect (p=1) at transect
- Area under the detection function curve = effective half-width μ
- Estimated from observed distances

Hierarchical distance sampling

- Unconditional detection probability
  $$P = \frac{\mu}{w}$$
- Observations *n* at each transect *j*
- $n_j \sim Binomial(N_j, P_j)$
- $N_j \sim Poisson(\lambda_j)$



Distance sampling in R with “unmarked”

- Create a desktop folder called “Lab 4”
- Open Rstudio, set the working directory to “Lab 4”, open a new script
- Install package “unmarked” and load it into workspace
- Look at general description of unmarked

```
> ?unmarked
```

Understanding basic detection functions

- Half-normal detection function
- Decline in detection with distance governed by scale parameter, σ
- $g(x) = \exp(-\frac{x^2}{2\sigma^2})$
- In R define sigma$= 50$
- Create a numeric vector, dist, with a range of distances, from 0 to 100, in 1-m increments
- Then calculate detection probability *p* (equal to *g*(*x*)) at each distance following the half-normal detection function
- Finally, plot *p* against dist using a line plot; label your axes; display y-axis from 0 to 1
- Repeat for sigma=30 and sigma=70, add detection functions to the same plot w/ different colors

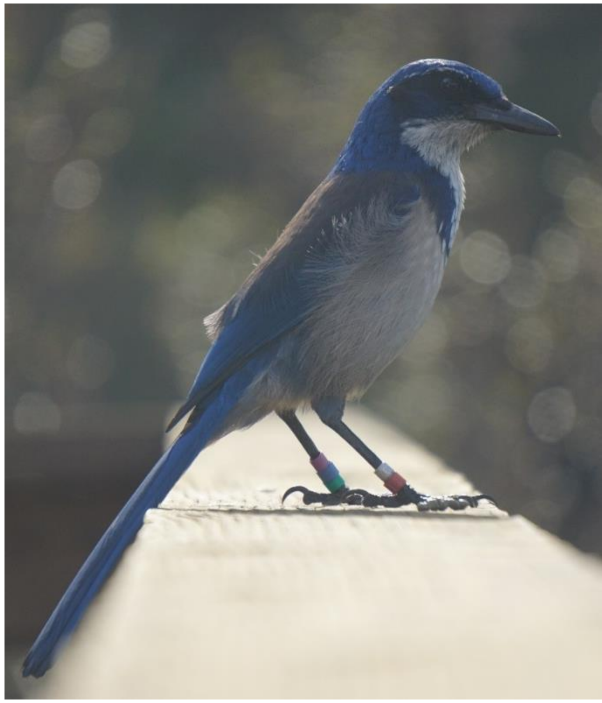
Understanding basic detection functions

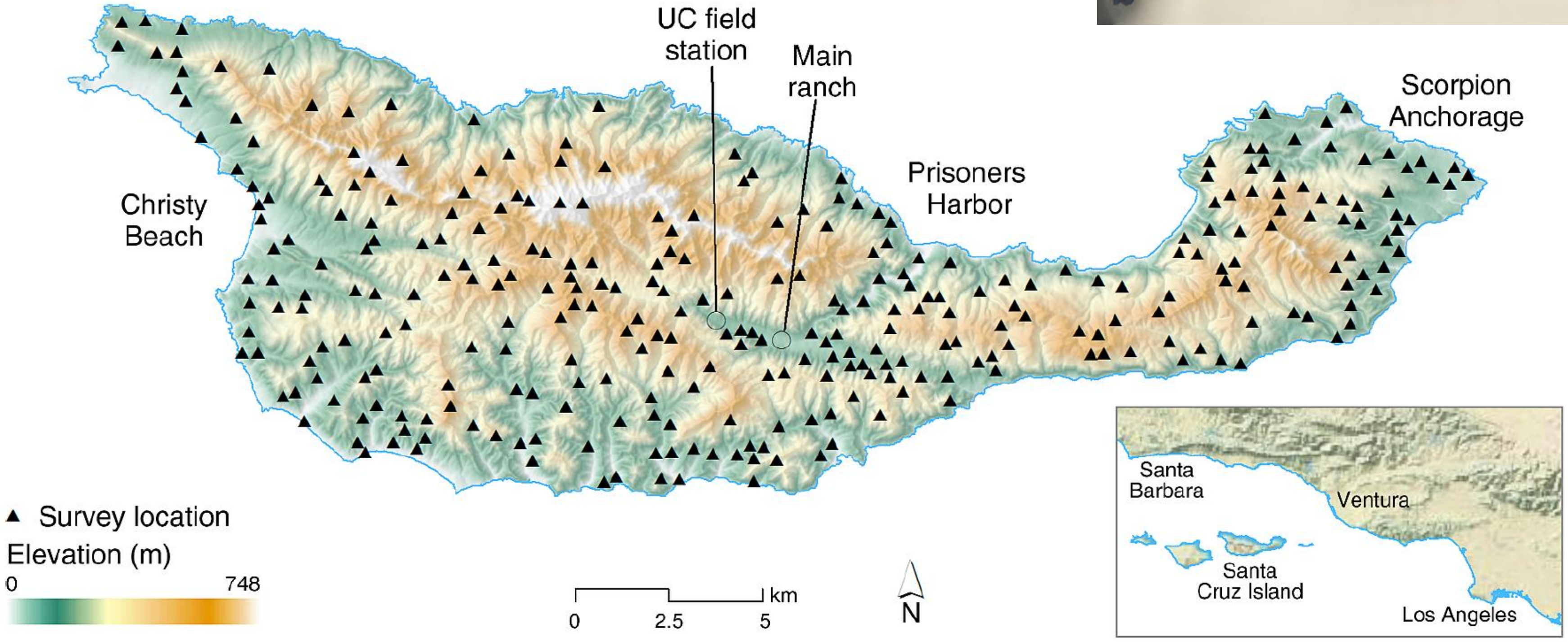
- Hazard detection function
- Decline in detection with distance governed by scale parameter, σ
- $g(x) = 1 - \exp(-\left(\frac{x}{\sigma}\right)^{-b})$

- Set sigma.haz $= 50$
- Set b = 2
- Then calculate detection probability p at each distance in dist following the hazard detection function
- Finally, plot p against dist using a line plot; label your axes; display y-axis from 0 to 1
- Repeat for b=1 and b=5, add detection probabilities to the same plot w/ different colors

The Island scrub-jay data set

- Distance sampling surveys of island scrub-jays on Santa Cruz Island, CA





The Island scrub-jay data set

- Data set is embedded in unmarked → load data

```
> data(issj)
```

- Look at structure of object “issj”
- Observations and covariates for 307 survey sites

```
$ issj[0-100]  : int  0 0 ...
$ issj(100-200]: int  0 0 ...
$ issj(200-300]: int  2 0 ...
```

- Covariates for each survey site

```
$ elevation
$ forest
$ chaparral
```

The Island scrub-jay data set

- Number of detections (across all sites) in each of the three distance bins

```
> sum(issj[,"issj[0-100]"])     #84
> sum(issj[,"issj(100-200]"]) #48
> sum(issj[,"issj(200-300]"]) #27
```

- Look at median and range of covariates using the summary() function, for example:

```
> summary(issj$elevation)
```

The Island scrub-jay data set

- Format data for analysis
- “unmarked” uses data frames for input data

```
>?distsamp
```

→ Data has to be an object of class “unmarkedFrameDS”

```
> ?unmarkedFrameDS


> umf <- unmarkedFrameDS(y=as.matrix(issj[,1:3]),
      siteCovs=data.frame(scale(issj[,6:8])),
      dist.breaks=c(0,100,200,300), unitsIn="m",
      survey="point")

> summary(umf)
```

The distsamp() function

- distsamp(formula, data, keyfun=c("halfnorm", "exp",

  "hazard", "uniform"), output=c("density", "abund"),

  unitsOut=c("ha", "kmsq"))
- Formula has two components, first for detection parameter, second for abundance
- ~1~1: no covariates on detection or abundance
- ~X~1: covariate X on detection, none on abundance
- ~1~X: no covariate on detection, X on abundance
- keyfun: shape of detection function
- output: abundance or abundance/area = density
- unitsOut: density per ha or per km2

Looking at model results

- Fit simplest model, no covariates, half-normal detection function

```
>hn<-distsamp(~1~1, umf, keyfun="halfnorm",
            output="density",unitsOut="ha")
```

```
> summary(hn)
```

- Parameters are reported on link scale (here, log)
- Back-transform to natural scale

```
> backTransform(hn, type="state")
```

```
> backTransform(hn, type="det")
```

- Now, fit the same model with a hazard detection function; save results in an object called haz

```
> haz<-distsamp(…)
```

- Report (back-transformed) sigma and density

Looking at model results

```
> summary(haz)
[…]
Density (log-scale):
 Estimate    SE     z P(>|z|)
    -1.63 0.726 -2.25  0.0247

Detection (log-scale):
 Estimate    SE   z  P(>|z|)
     3.86 0.654 5.9 3.59e-09     log(sigma.haz)

Hazard-rate(scale) (log-scale):
 Estimate    SE    z P(>|z|)
     0.76 0.234 3.24 0.00118     log(b)
```

Comparing models by AIC

- Create a fitList object

```
> dsfitlist<- fitList(hn,haz)
```

- Compare AIC for both models

```
> modSel(dsfitlist)
```

- Get all estimates of a distance sampling model on the link scale using `model@estimates`

Including covariates

- Start by including chaparral into the abundance model

```
> haz.chap<-distsamp(~1~chaparral, umf,
            keyfun="hazard", output="density",
            unitsOut="ha")

> haz.chap
```

- Use parameter estimates to calculate expected log(density), then backtransform
- Similar to GLM, ”predict” does both for you

```
> pred.chap<-predict(haz.chap, "state")
```

- To plot expected density against chaparral, we need to sort predictions by ascending order of chaparral

```
> s.in<-order(issj$chaparral)

> plot(issj$chaparral[s.in], pred.chap[s.in,1],
      type="l", xlab="Chaparral",
      ylab="Exp. abundance")
```

Including covariates

- Next, build a new model including chaparral as a covariate on detection only
- Compare that model with haz.chap (chaparral on abundance) by AIC
- Which model fits better?
- What does that tell us?

- Then, build two more models, with elevation as a covariate on abundance, on detection
- Compare these models, and the hazard model without covariates, using AIC
- Which covariate is important for which parameter?