Hierarchical Distance Sampling

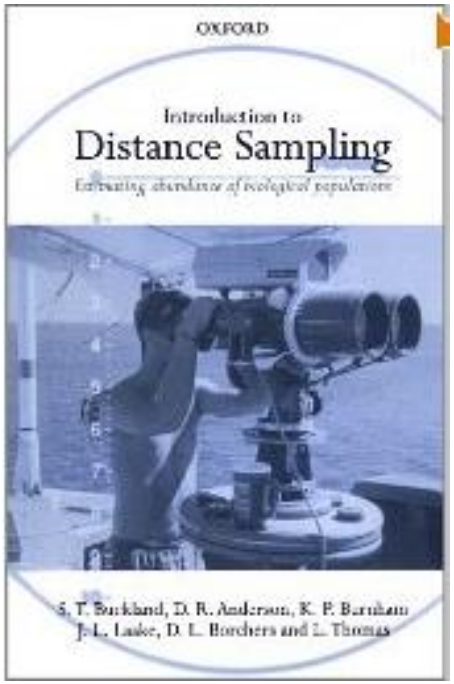
Applied Hierarchical Modeling Workshop

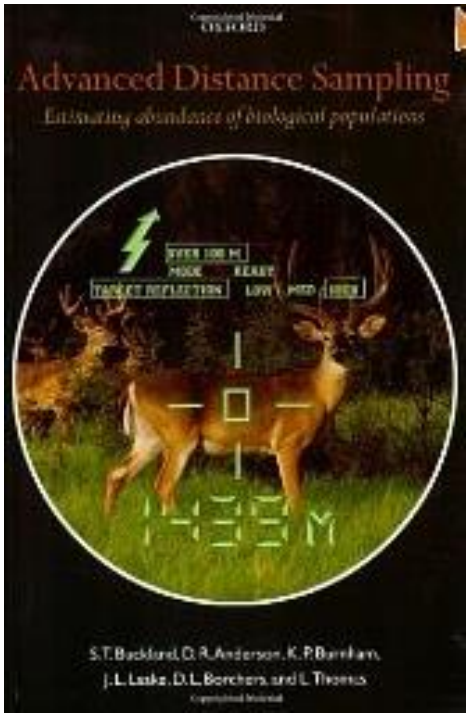
Patuxent Wildlife Research Center

Nov 2015

Distance Sampling

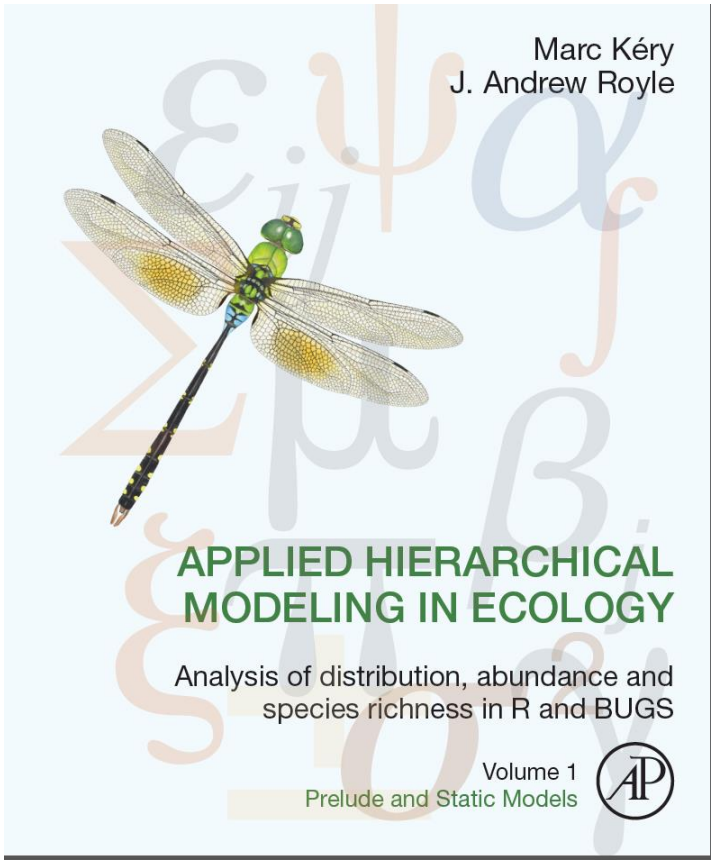
- Two excellent books written on the topic





Distance Sampling

- Or maybe three



Conventional Distance Sampling

Basic Idea:

- Estimate density/abundance by modeling detection probability as a function of distance.

Features:

- Population size can be estimated from single survey
- Explicit link between population size and density

Conventional Distance Sampling

$$N = \frac{n}{p} \qquad \text{E(n) = p*N}$$

Modeling Detection Probability

- Choose a detection function, $g(x)$
  - Assume $g(0) = 1$
  - Half-normal, hazard rate, exponential, etc.
- Specify a probability distribution for distances $[x]$ typically based on uniform distribution of object ***locations***
  - For line transects, $[x] = \frac{1}{w}$
  - For point counts, $[x] = \frac{2x}{w^2}$
- The average is $p = \int_0^w g(x)[x]dx$

w = count distance

Conventional Distance Sampling

Key idea:

- The average is $p = \int_0^w g(x)[x]dx$

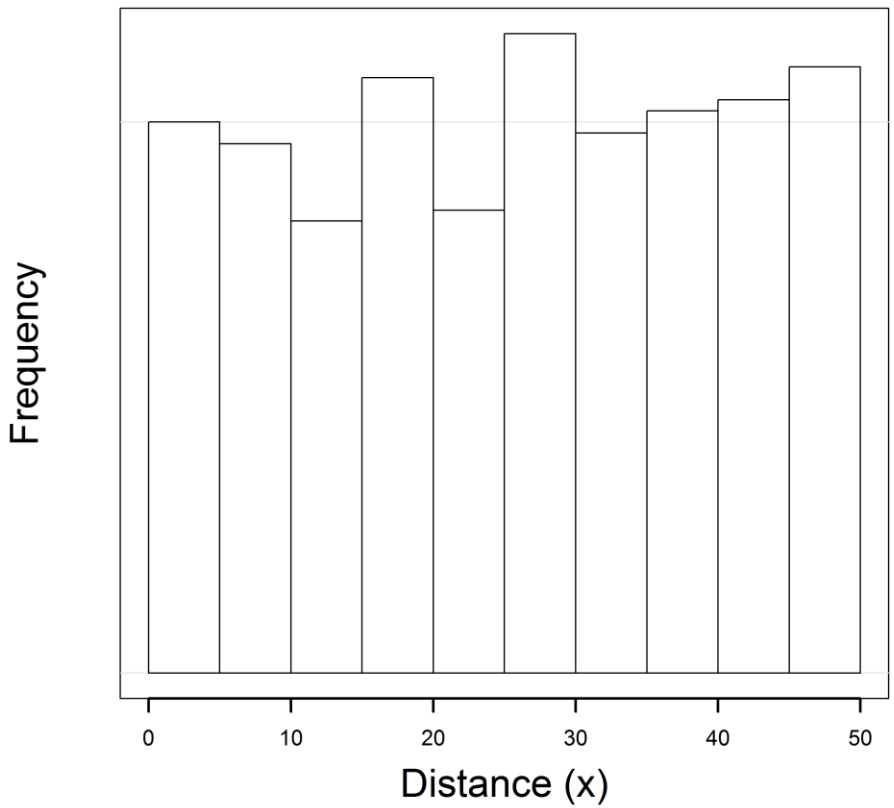
  [examples to follow]

Therefore CDS involves two explicit models:

- "observation model" $\Pr(y = 1) = g(x)$
- "process model", $[x]$
  - In what sense is the distribution of distances an ecological process???

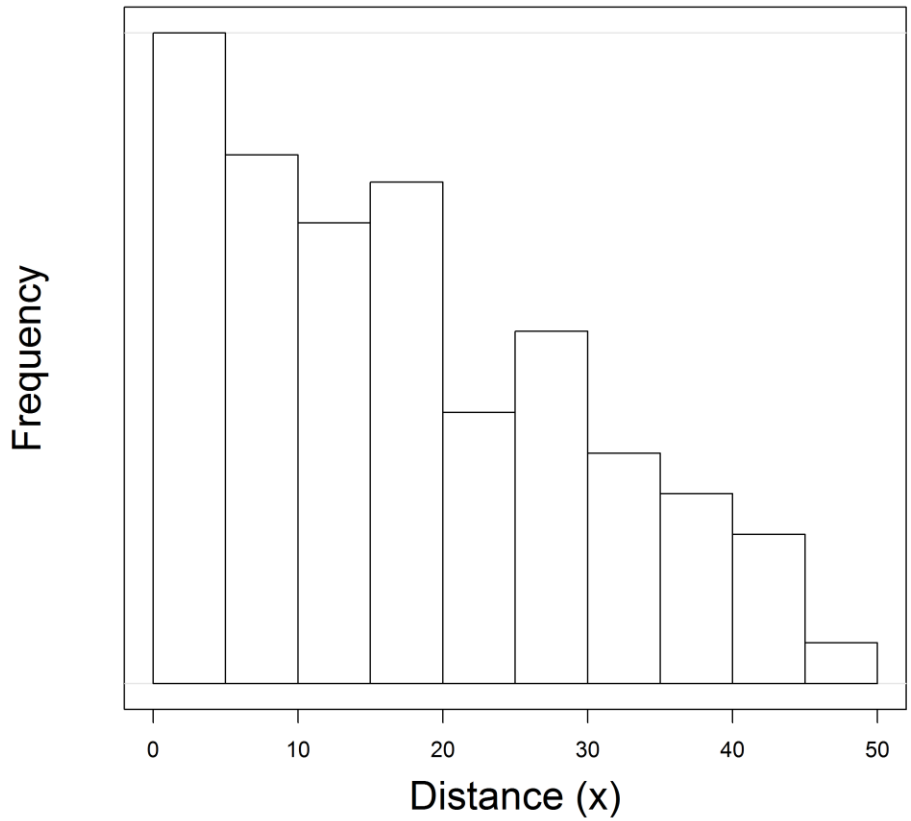
Computing p, average detection probability

Realization of “true” distances

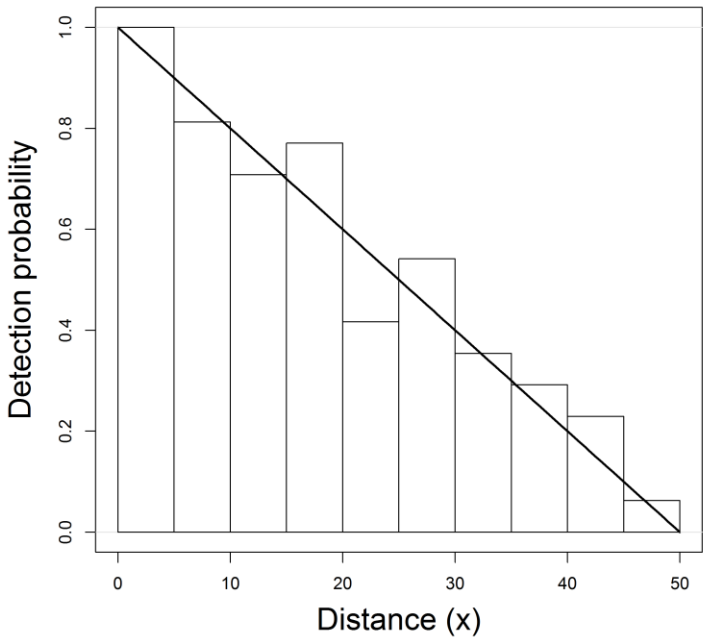


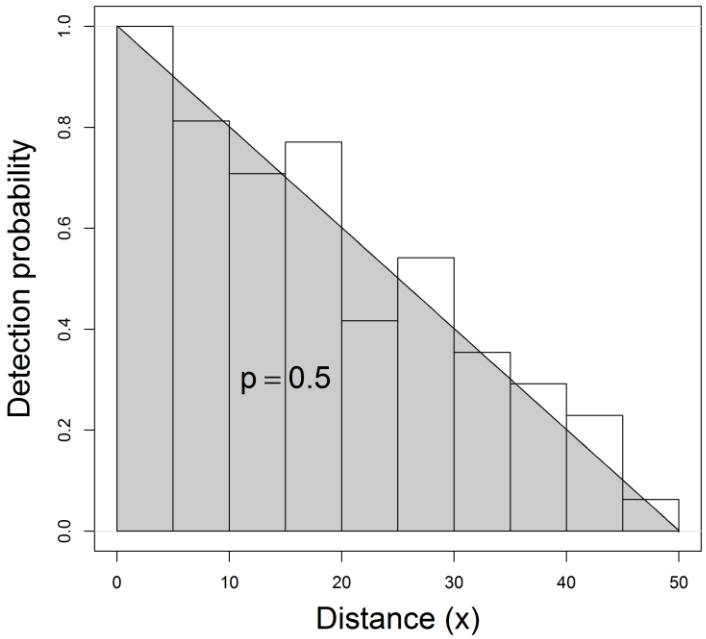
Computing p, average detection probability

Observed distances

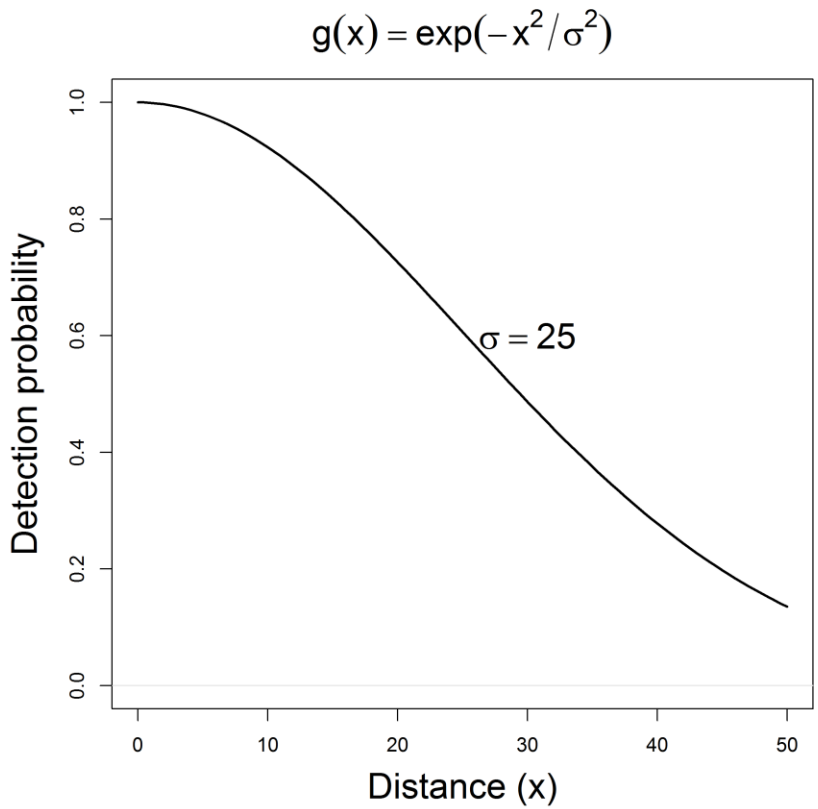


Computing p, average detection probability

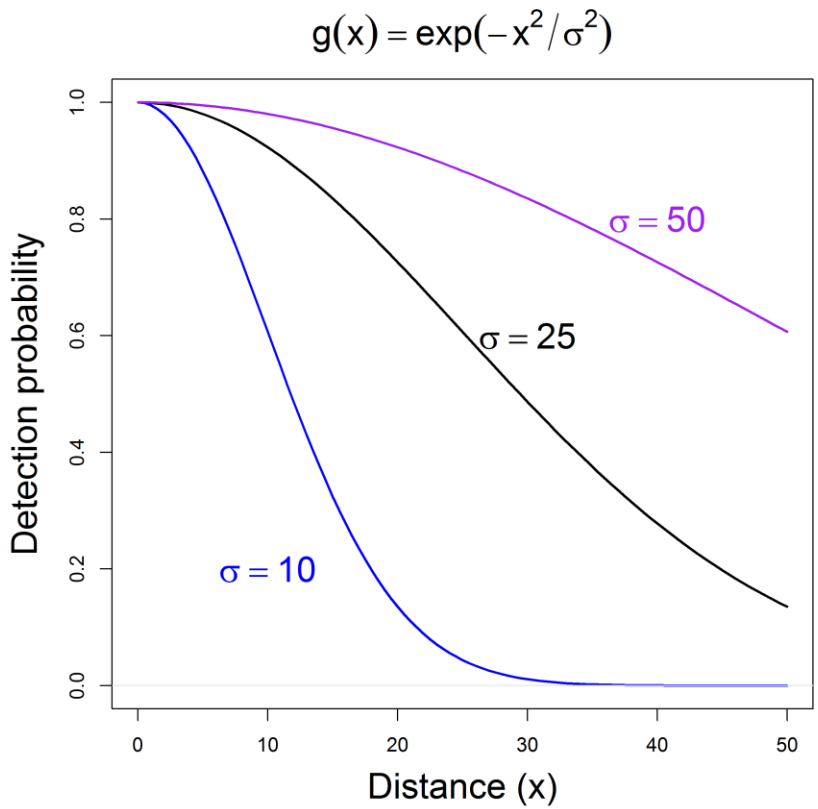




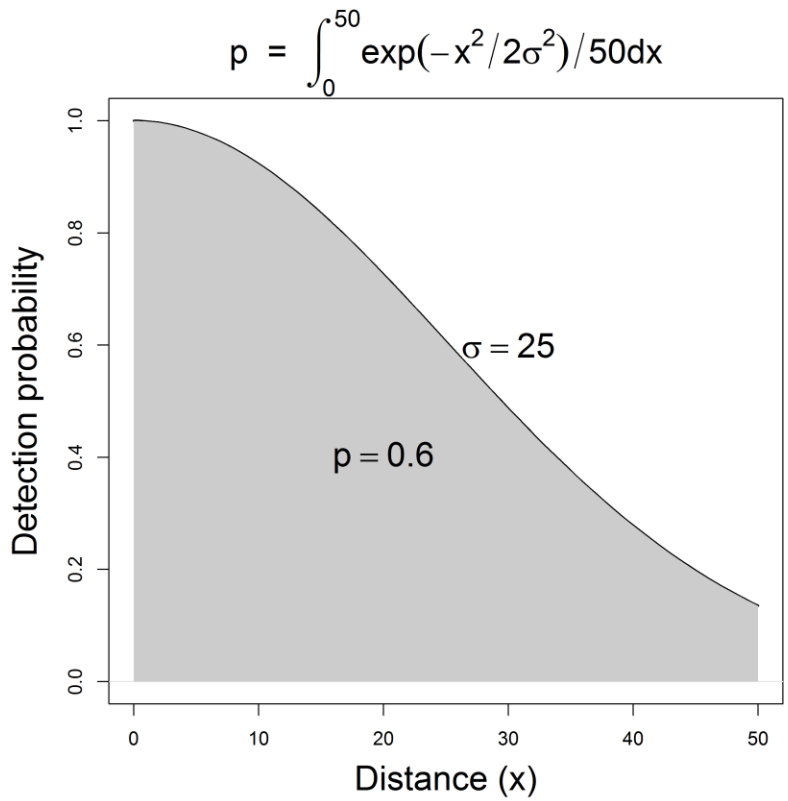
Computing p, average detection probability



Computing p, average detection probability



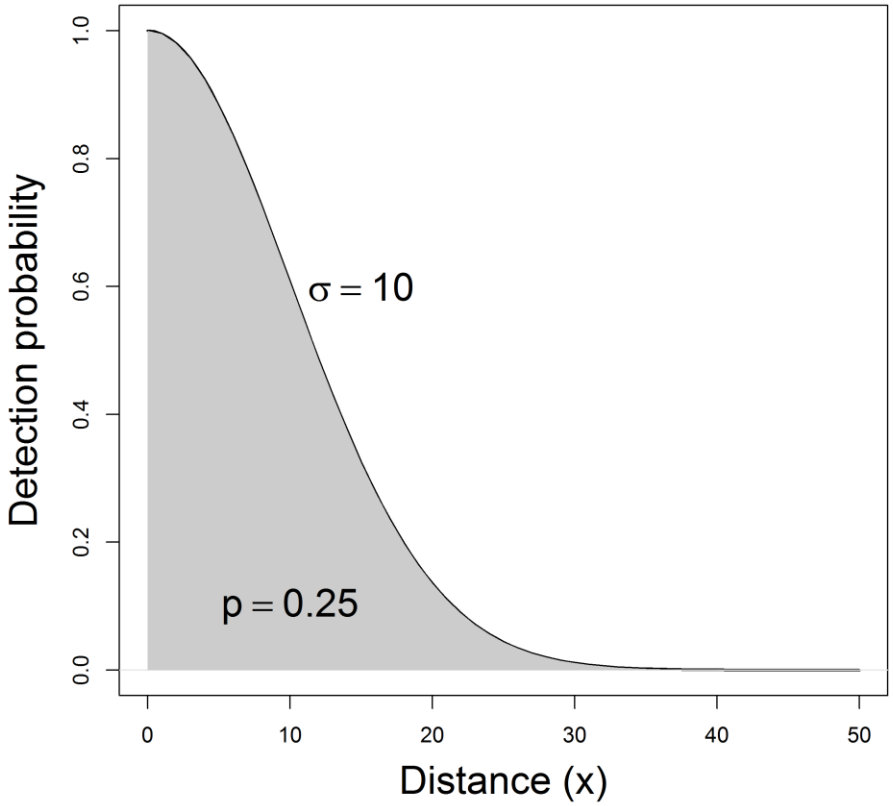
Computing p, average detection probability



p is related to σ

Computing p, average detection probability

p is related to σ



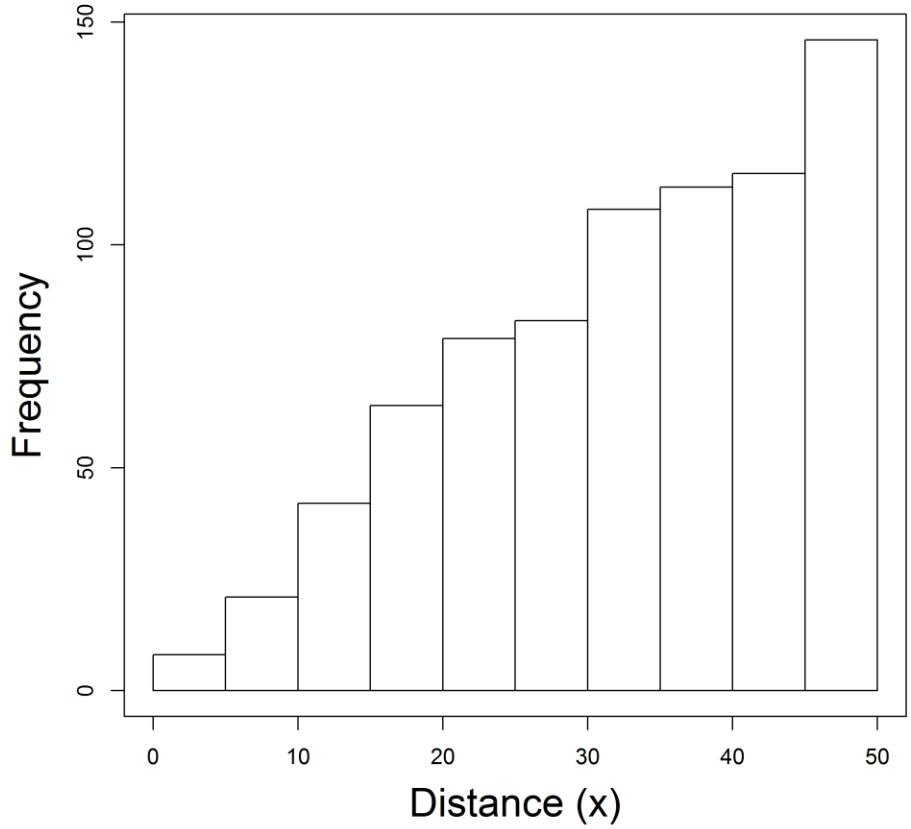
What about point counts?

- Slim chance of an individual being close to a point (not much area in the vicinity of the point)
  - $g(x)$ is the same (half-normal, etc..)
  - But $[x]$ is different
  - $[x] = 2x/w^2$

Triangular distribution: assumes uniform distribution of objects

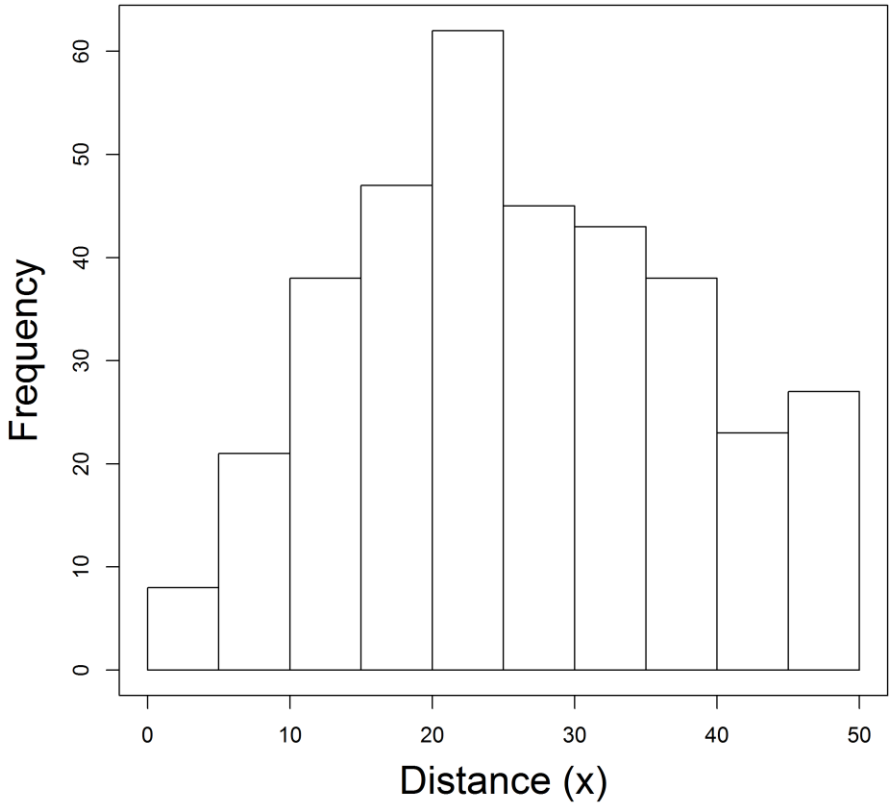
What about point counts?

Realization of “truth”



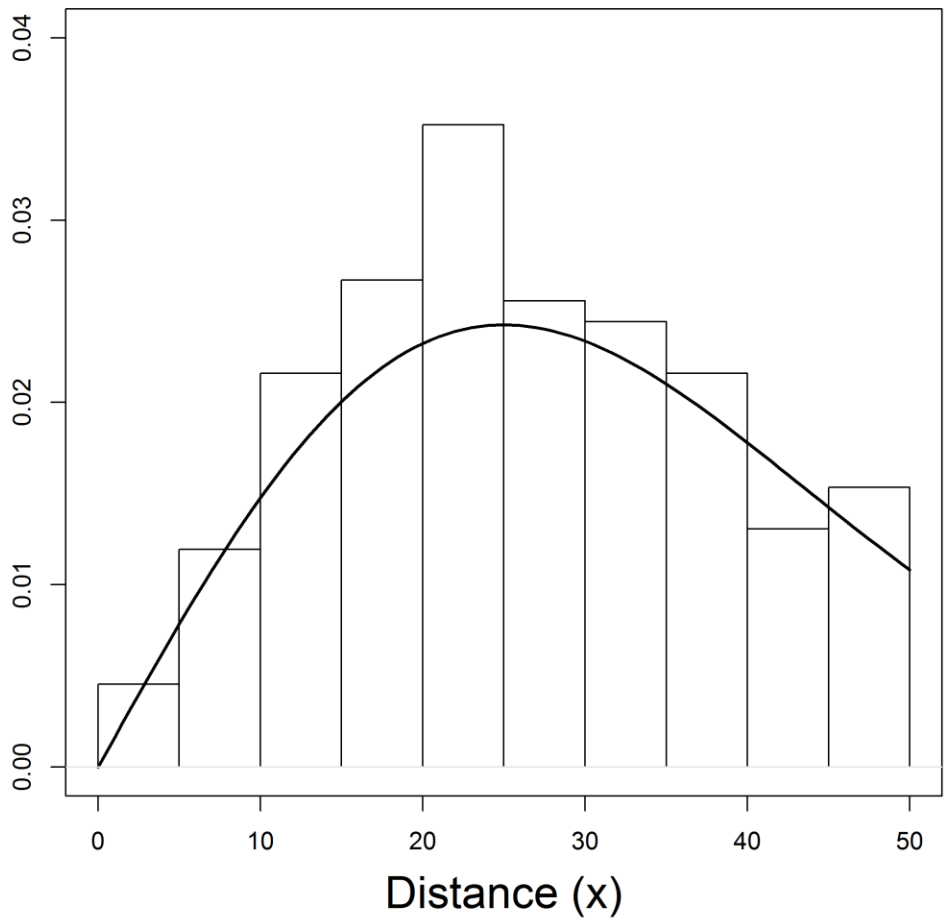
What about point counts?

Observed distances from count location



What about point counts?

Distribution of observed distances is $g(x)$weighted by increasing area



Estimation

- Fit this area-weighted detection model to the observed distances and then integrate the curve.

Non-conventional Distance Sampling

Some important extensions

- $g(0) < 1$
- Animals avoid (or attracted to) the transect
- Spatial variation in abundance

Spatial Variation in Abundance

- Modeling variation among transects or points (Royle et al. 2004)

Ecology, 85(6), 2004, pp. 1591–1597
© 2004 by the Ecological Society of America

MODELING ABUNDANCE EFFECTS IN DISTANCE SAMPLING

J. Andrew Royle,[1,4] Deanna K. Dawson,[2] and Scott Bates[3]

[1]*U.S. Fish and Wildlife Service, Division of Migratory Bird Management, 11510 American Holly Drive, Laurel, Maryland 20708 USA*
[2]*USGS Patuxent Wildlife Research Center, 12100 Beech Forest Road, Laurel, Maryland 20708 USA*
[3]*National Park Service, Center for Urban Ecology, 4598 MacArthur Boulevard, NW, Washington, D.C. 20007-4227 USA*

- Modeling variation within transects or points

- Both

Hierarchical Distance Sampling

- Data
  - "binned data" collected at M point or line transects
  - y_{ij} = count in distance class j of transect i

Transect	dclass 1 [0-50m]	dclass 2 (50-100m]	dclass 3 (100-200m]	covariate1	covariate2
transect 1	2	0	1	v_{11}	v_{12}
transect 2	3	0	0	v_{21}	v_{22}
transect 3	2	1	1	v_{31}	v_{32}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
transect R	4	2	0	v_{R1}	v_{R2}

Hierarchical Distance Sampling

- **Distance sampling observation model**

$$(y_{i1}, \dots, y_{iJ}) \sim \text{Multinomial}(N_i, (\pi_{ij}))$$

where π_{ij} is the multinomial cell probability for distance class j and sample unit i – these depend on σ

Covariates on detection:

$$\log(\sigma_i) = \alpha_0 + \alpha_1 * covariate_i$$

- **Abundance model**

$$N_i \sim Poisson(\lambda_i)$$

$$\log(\lambda_i) = \beta_0 + \beta_1 * Covariate_i$$

Multinomial distribution

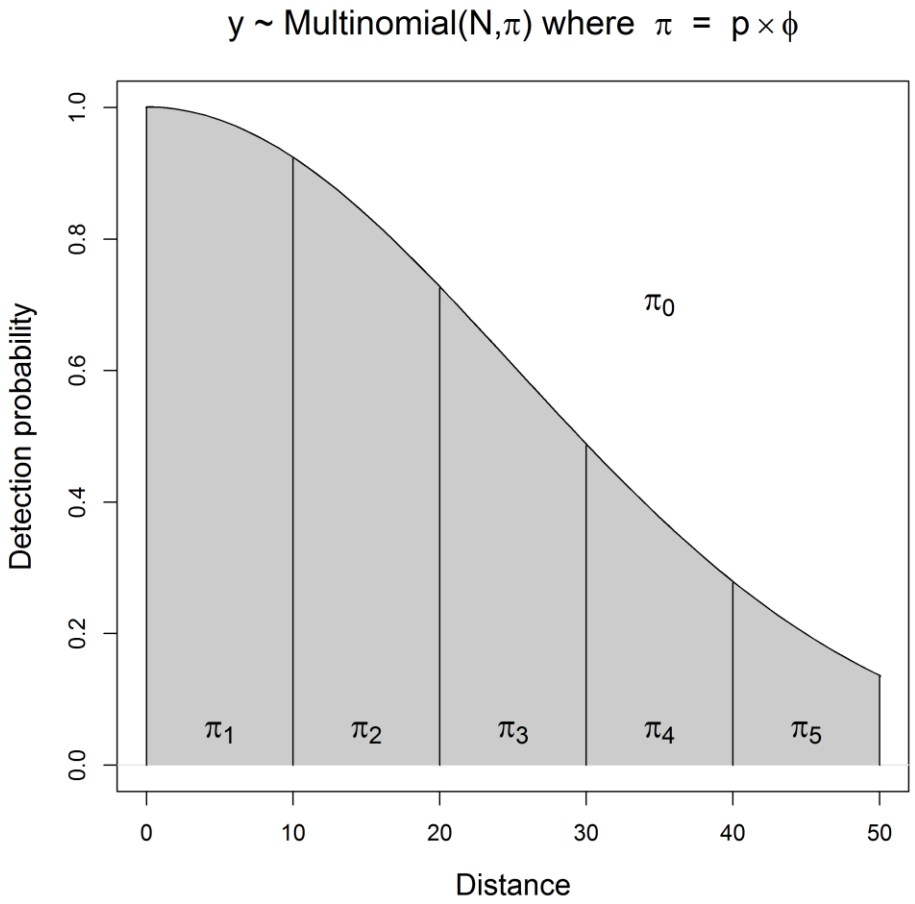
- Simulate data with these cell probabilities

```
N <- 10
y <- as.vector(rmultinom(1, N, c(0.19,0.17,0.12,0.08,0.04,0.4)))
names(y) <- c("[0-10]","(10-20]","(20-30]","(30-40]","(40-
50]","missed")

y
[0-10] (10-20] (20-30] (30-40] (40-50]  missed
     3       1       3       0       0       3
```

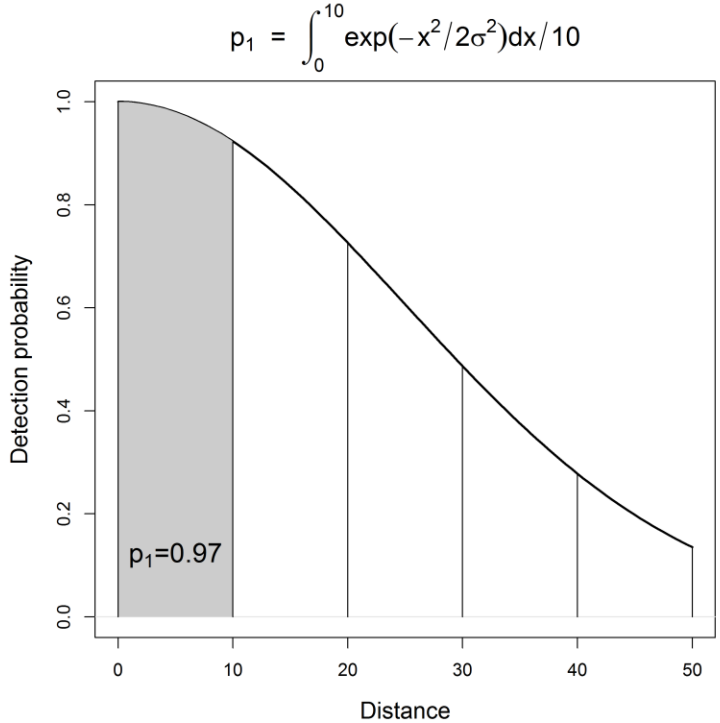
Computing multinomial cell probabilities

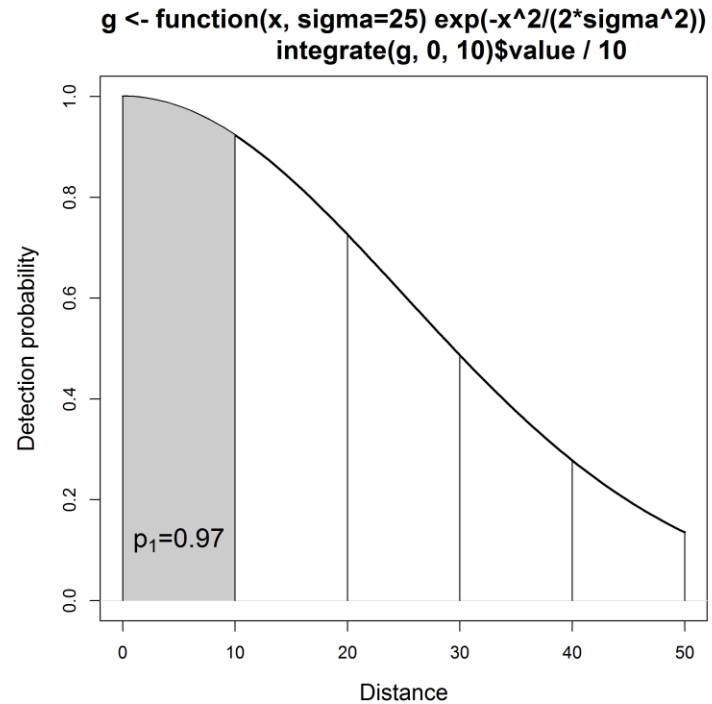
- Multinomial cell probability for distance class h, π_h, is the expected fraction of observed distances in class h.
- Calculation of π_h: proportion of the population in class h (ψ_h) multiplied by the average probability of detection in that class (p_h)
- For line transects:
  - $\psi_h = b_h/B$ (width of bin h over transect width)
  - Average p = $\int_x g(x)\,\Pr(x \mid class\ h)\,dx$
  - Can approximate this sometimes by $g(x_h)$ where x_h is the mid-point of the interval h

Computing multinomial cell probabilities

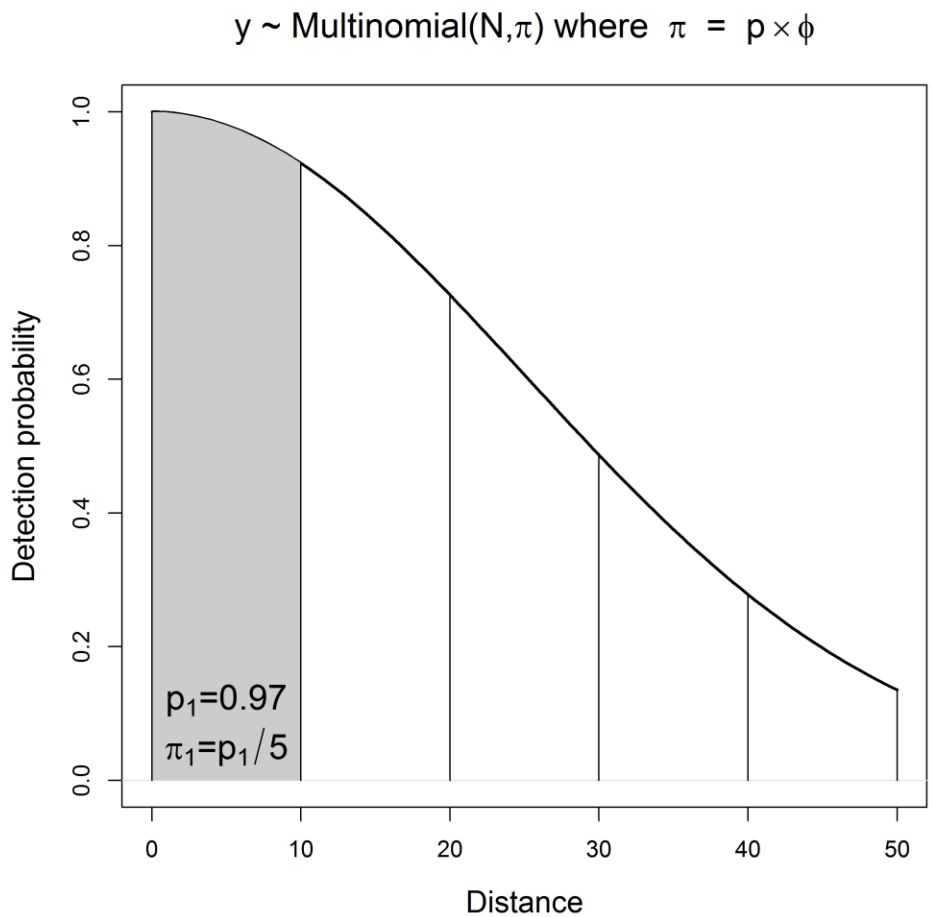


Computing multinomial cell probabilities

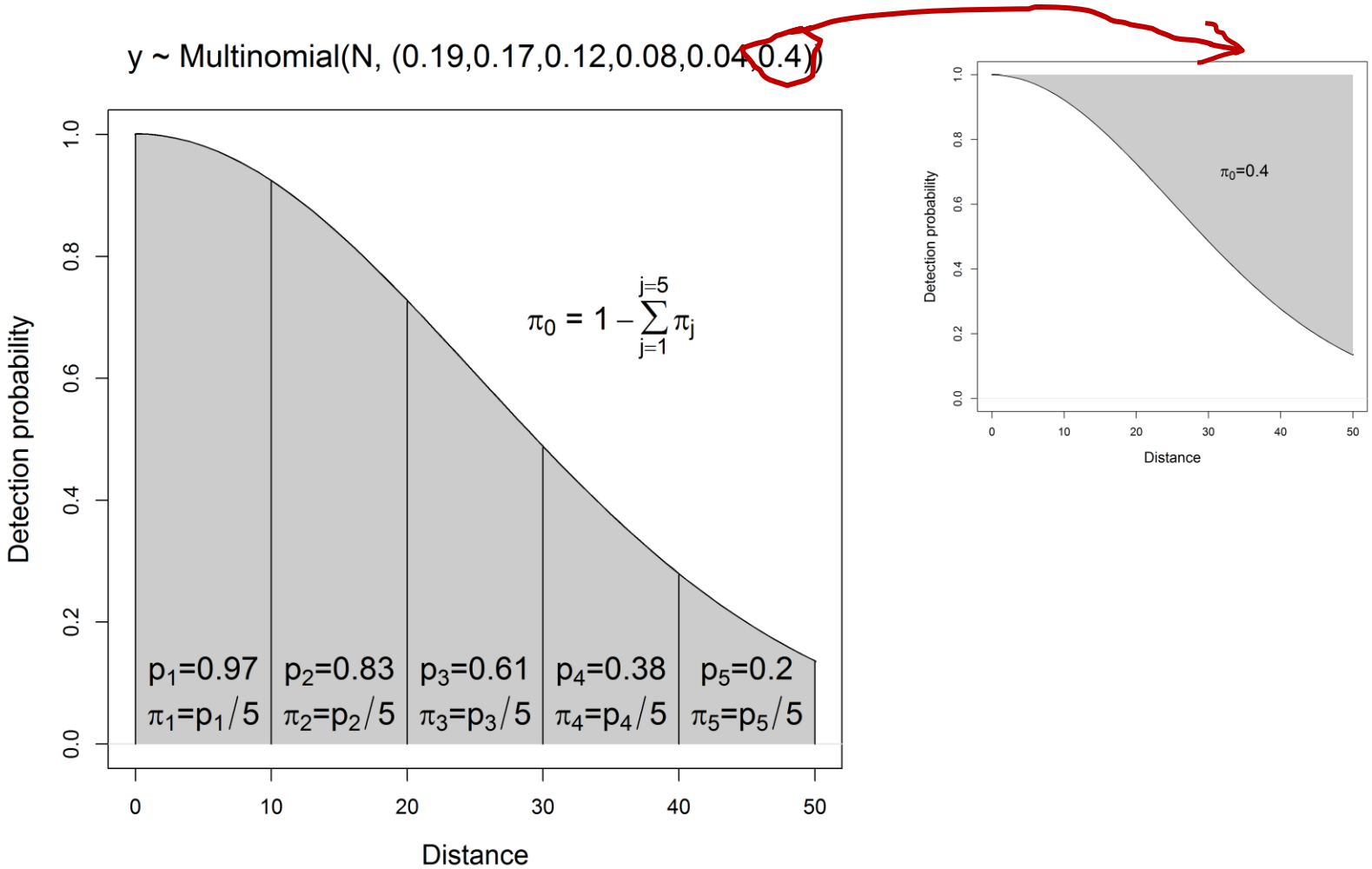




Computing multinomial cell probabilities



Computing multinomial cell probabilities



unmarked

Functions for fitting HDS models in `unmarked`

- `distsamp` – basic "closed population" hierarchical distance sampling. Only uses $N \sim Poisson$

```
distsamp(formula, data, keyfun=c("halfnorm", "exp", "hazard",
"uniform"),output=c("density", "abund"), unitsOut=c("ha", "kmsq"),
starts, ...)
```

- `gdistsamp` – for open populations. $N[i,t]$ is dynamic – "robust design". Also allows for other abundance distributions. Right now just negative binomial.

```
gdistsamp(lambdaformula, phiformula, pformula, data,
        keyfun =c("halfnorm", "exp", "hazard", "uniform"),
        output = c("abund","density"), unitsOut = c("ha", "kmsq"),
        mixture = c("P", "NB"), K, starts, method = "BFGS",
        se = TRUE, rel.tol=1e-4, ...)
```

unmarked

Constructor functions :

`unmarkedFrameDS` and `unmarkedFrameGDS`

```
umf <- unmarkedFrameDS(y=y, siteCovs=site.covs,
dist.breaks=db, survey="point", unitsIn="m")
```

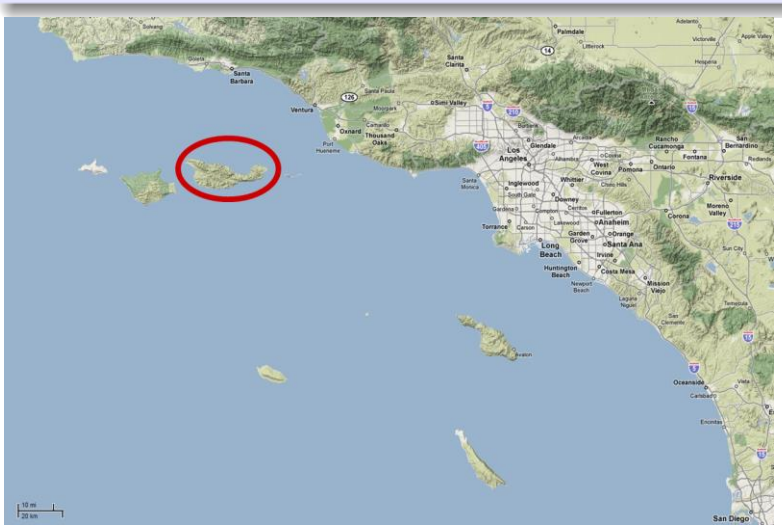
Open populations:

- `unmarkedMultFrame(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs)`
- `unmarkedFrameGMM(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs, type, obsToY, piFun)`
- `unmarkedFrameGDS(y, siteCovs, numPrimary, yearlySiteCovs, dist.breaks, survey, unitsIn, tlength)`
- `unmarkedFrameGPC(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs)`

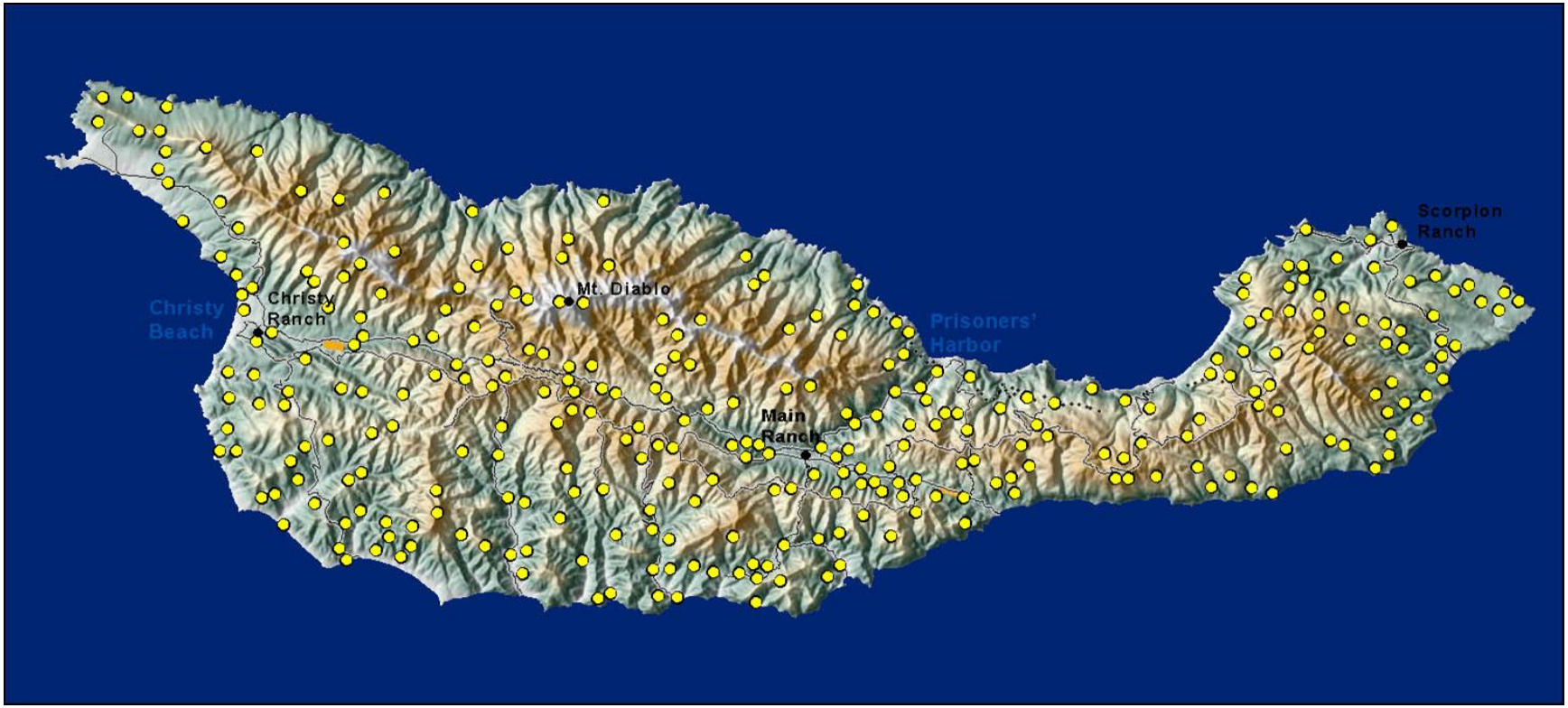
Application: The ISSJ

Estimating population size of Island Scrub Jay

- Endemic to Santa Cruz Island, California
- 307 point count locations, birds counted in a 300m circle (Sillett et al., 2012).
- Distance binned to 100 m classes because close-by birds were responding to the observer.
- Objectives
  - How many?
  - Make a map of $E[N]$
  - Predict under alternative/historical landscapes



Sillett et al. ISSJ survey



Sillett et al. ISSJ survey

Organizing the data into an unmarkedFrame:

```
# Load, view and format the ISSJ data
library(unmarked)
data(issj)
round(head(issj), 2)
 issj[0-100] issj(100-200] issj(200-300]   x       y elevation forest chaparral
1           0           0           2 234870.1 3767154     51.39   0.02      0.24
2           0           0           0 237083.0 3766804    156.88   0.01      0.47
3           0           0           0 235732.0 3766717    144.81   0.02      0.77
4           0           0           0 237605.0 3766719    184.27   0.26      0.21
5           0           0           0 234239.1 3766570    111.35   0.00      0.00
6           0           0           0 235005.1 3766420    204.13   0.16      0.34

# Package things up into an unmarkedFrame
covs <- issj[,c("elevation", "forest", "chaparral")]
area <- pi*300^2 / 100^2              # Area in ha
jayumf <- unmarkedFrameDS(y=as.matrix(issj[,1:3]),
   siteCovs=data.frame(covs, area),
   dist.breaks=c(0, 100, 200, 300),
   unitsIn="m", survey="point")
```

Sillett et al. ISSJ survey

```
# Fit model 1
(fm1 <- distsamp(~chaparral ~chaparral + elevation +
offset(log(area)), jayumf, keyfun="halfnorm", output="abund"))

Call:
distsamp(formula = ~chaparral ~ chaparral + elevation +
offset(log(area)), data = jayumf, keyfun = "halfnorm", output =
"abund")

Abundance:
            Estimate      SE      z  P(>|z|)
(Intercept) -3.50982 0.31261 -11.23 2.99e-29
chaparral    4.11503 0.62458   6.59 4.44e-11
elevation   -0.00216 0.00073  -2.96 3.11e-03

Detection:
                 Estimate    SE     z   P(>|z|)
sigma(Intercept)     5.02 0.161 31.15 5.65e-213
sigmachaparral      -1.07 0.319 -3.36  7.73e-04

AIC: 964.7203
```

Sillett et al. ISSJ survey

```
# Fit model 2

(fm2 <- distsamp(~1 ~chaparral + elevation + offset(log(area)),
    jayumf, keyfun="halfnorm", output="abund"))

Call:
distsamp(formula = ~1 ~ chaparral + elevation + offset(log(area)),
    data = jayumf, keyfun = "halfnorm", output = "abund")

Abundance:
            Estimate       SE      z  P(>|z|)
(Intercept) -2.71972 0.200946 -13.53 9.77e-42
chaparral    2.12760 0.309172   6.88 5.92e-12
elevation   -0.00212 0.000728  -2.91 3.59e-03

Detection:
 Estimate     SE    z P(>|z|)
     4.58 0.0488 93.9       0

AIC: 976.2306
```

Sillett et al. ISSJ survey

```
(pb <- parboot(fm1, fitstats, nsim=1000, report=5))
(c.hat <- pb@t0[2] / mean(pb@t.star[,2]))  # c-hat as ratio of observed
                            # and mean of expected value of Chi2 (under H0)
                            # (see, e.g., Johnson et al., Biometrics, 2010)
   Chisq
2.590553

residuals(fm1)              # Can inspect residuals
plot(pb)                    # Not shown
print(pb)

Call: parboot(object = fm1, statistic = fitstats, nsim = 1000, report = 5)

Parametric Bootstrap Statistics:
               t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE           421          262.7             16.5            0
Chisq        2357         1447.1             66.1            0
freemanTukey  210           42.9             10.1            0

t_B quantiles:
              0% 2.5% 25% 50% 75% 97.5% 100%
SSE          110  130 147 158 169   193  213
Chisq        739  806 864 903 950  1056 1364
freemanTukey 131  147 161 167 174   187  202
```

Sillett et al. ISSJ survey

Make the model more complex models, more covariates. First let's standardize covariates and add area as an offset:

```
# Standardize the covariates
sc <- siteCovs(jayumf)
sc.s <- scale(sc)
sc.s[,"area"] <- pi*300^2 / 10000  # Don't standardize area
siteCovs(jayumf) <- sc.s
summary(jayumf)
```

Sillett et al. ISSJ survey

```
# Fit a bunch of models and produce a model selection table.
fall <- list()   # make a list to store the models

# With the offset output=abund is the same as output = density
fall$Null <- distsamp(~1 ~offset(log(area)), jayumf, output="abund")
fall$Chap. <- distsamp(~1 ~chaparral + offset(log(area)), jayumf,
output="abund")
fall$Chap2. <- distsamp(~1 ~chaparral+I(chaparral^2)+offset(log(area)),jayumf,
output="abund")
fall$Elev. <- distsamp(~1 ~ elevation+offset(log(area)), jayumf,
output="abund")
fall$Elev2. <- distsamp(~1 ~ elevation+I(elevation^2)+offset(log(area)),
    jayumf, output="abund")
fall$Forest. <- distsamp(~1 ~forest+offset(log(area)), jayumf,
   output="abund")
fall$Forest2. <- distsamp(~1 ~forest+I(forest^2)+offset(log(area)),
    jayumf, output="abund")
fall$.Forest <- distsamp(~forest ~offset(log(area)), jayumf,
    output="abund")
fall$.Chap <- distsamp(~chaparral ~offset(log(area)), jayumf,
    output="abund")
fall$C2E. <- distsamp(~1 ~ chaparral + I(chaparral^2) + elevation +
    offset(log(area)),jayumf, output="abund")
fall$C2F2. <- distsamp(~1 ~chaparral + I(chaparral^2) + forest +
    I(forest^2)+offset(log(area)), jayumf,  output="abund")
fall$C2E.F <- distsamp(~forest ~chaparral+I(chaparral^2)+elevation+
    offset(log(area)), jayumf, output="abund")
fall$C2E.C <- distsamp(~chaparral ~chaparral + I(chaparral^2) + elevation +
    offset(log(area)), jayumf, output="abund")
```

Sillett et al. ISSJ survey

```
# Create a fitList and a model selection table
(msFall <- modSel(fitList(fits=fall)))

         nPars     AIC delta   AICwt cumltvWt
C2E.C        6  951.35  0.00 9.9e-01     0.99
C2E.         5  961.01  9.66 7.9e-03     1.00
C2E.F        6  962.95 11.60 3.0e-03     1.00
Chap2.       4  965.95 14.60 6.7e-04     1.00
C2F2.        6  968.13 16.78 2.2e-04     1.00
Chap.        3  981.39 30.04 3.0e-07     1.00
.Chap        3 1007.02 55.67 8.1e-13     1.00
Forest2.     4 1015.07 63.72 1.4e-14     1.00
Elev2.       4 1017.33 65.98 4.7e-15     1.00
Elev.        3 1018.10 66.75 3.2e-15     1.00
Null_D       2 1018.12 66.77 3.1e-15     1.00
Null         2 1018.12 66.77 3.1e-15     1.00
Forest.      3 1019.65 68.30 1.5e-15     1.00
.Forest      3 1020.08 68.73 1.2e-15     1.00
```

```
# Check out the best model
fall$C2E.C

Call:
distsamp(formula = ~chaparral ~ chaparral + I(chaparral^2) +
    elevation + offset(log(area)), data = jayumf, output = "abund")

Abundance:
               Estimate     SE      z  P(>|z|)
(Intercept)      -2.562 0.1589 -16.12 1.75e-58
chaparral         1.230 0.1602   7.68 1.64e-14
I(chaparral^2)   -0.282 0.0775  -3.64 2.68e-04
elevation        -0.238 0.0926  -2.57 1.02e-02

Detection:
                 Estimate     SE     z  P(>|z|)
sigma(Intercept)    4.686 0.0682 68.75 0.000000
sigmachaparral     -0.208 0.0626 -3.32 0.000892

AIC: 951.3504
```

```
# Check out the goodness-of-fit of this model
(pb.try2 <- parboot(fall$C2E.C, fitstats, nsim=1000, report=5))
Call: parboot(object = fall$C2E.C, statistic = fitstats, nsim = 1000, report =
5)

Parametric Bootstrap Statistics:
               t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE           425          267.7            16.13           0
Chisq        2197         1285.9            70.54           0
freemanTukey  207           43.4             9.76           0

t_B quantiles:
              0% 2.5% 25% 50% 75% 97.5% 100%
SSE          112  128 147 156 168   190  233
Chisq        752  794 864 904 946  1057 1419
freemanTukey 134  144 157 164 170   183  195



# Express the magnitude of lack of fit by an overdispersion factor
(c.hat <- pb.try2@t0[2] / mean(pb.try2@t.star[,2]))  #    Chisq
2.411948
```

Sillett et al. ISSJ survey

Fit some NB models using `gdistsamp`:

```
covs <- issj[,c("elevation", "forest", "chaparral")]
area <- pi*300^2 / 100^2                # Area in ha
jayumf <- unmarkedFrameGDS(y=as.matrix(issj[,1:3]),
   siteCovs=data.frame(covs, area), numPrimary=1,
   dist.breaks=c(0, 100, 200, 300),
   unitsIn="m", survey="point")
sc <- siteCovs(jayumf)
sc.s <- scale(sc)
sc.s[,"area"] <- pi*300^2 / 10000  # Don't standardize area
siteCovs(jayumf) <- sc.s
summary(jayumf)
```

```
# Fit the model using gdistsamp and look at the fit summary

(nb.C2E.C <- gdistsamp( ~chaparral + I(chaparral^2) + elevation +
   offset(log(area)), ~1, ~chaparral, data =jayumf, output="abund",
   mixture="NB", K = 150))

gdistsamp(lambdaformula = ~chaparral + I(chaparral^2) + elevation +
    offset(log(area)), phiformula = ~1, pformula = ~chaparral,
    data = jayumf, output = "abund", mixture = "NB", K = 150)

Abundance:
               Estimate    SE      z  P(>|z|)
(Intercept)      -2.516 0.198 -12.73 4.17e-37
chaparral         1.432 0.229   6.25 4.01e-10
I(chaparral^2)   -0.376 0.114  -3.28 1.04e-03
elevation        -0.227 0.146  -1.55 1.20e-01

Detection:
            Estimate     SE     z  P(>|z|)
(Intercept)    4.679 0.0658 71.14 0.000000
chaparral     -0.199 0.0600 -3.32 0.000905

Dispersion:                          # Note the NB dispersion parameter
 Estimate    SE     z  P(>|z|)       # Scale is log(tau)
   -1.02 0.215 -4.73 2.23e-06

AIC: 695.4445
```

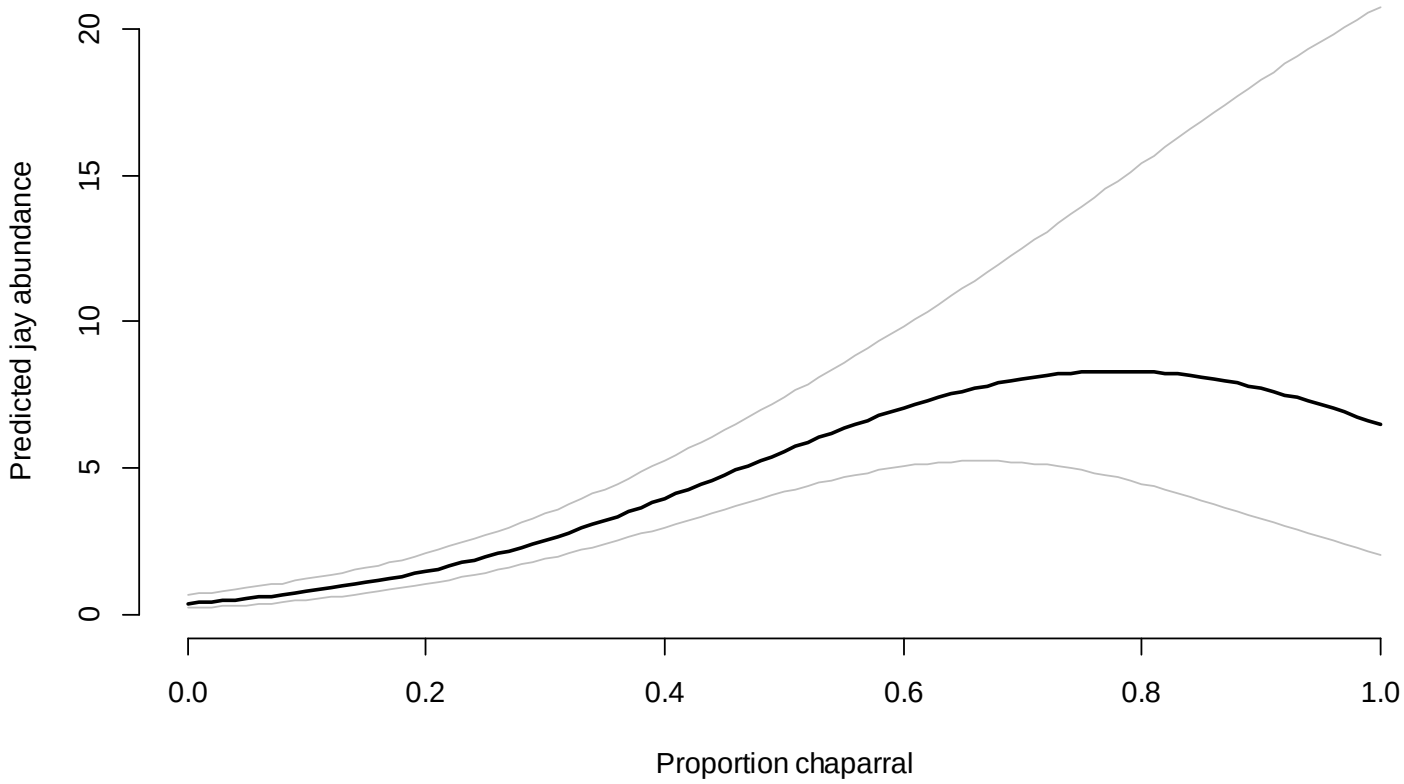
Goodness-of-fit assessment using `parboot`:

```
(pb.try3 <- parboot(nb.C2E.C, fitstats, nsim=1000, report=5))

Call: parboot(object = nb.C2E.C, statistic = fitstats, nsim = 1000,
report = 5)

Parametric Bootstrap Statistics:
               t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
SSE           430           97.4             99.7     0.143856
Chisq        2200          868.1            161.2     0.000999
freemanTukey  211           20.8             20.9     0.159840

t_B quantiles:
              0% 2.5%  25%  50%  75% 97.5% 100%
SSE          117  188  265  315  379   576 1133
Chisq        921 1073 1225 1313 1425  1706 2556
freemanTukey 113  150  177  190  205   234  260

There were 50 or more warnings (use warnings() to see the first 50)

(c.hat <- pb.try3@t0[2] / mean(pb.try3@t.star[,2]))  #
   Chisq
1.65186
```

Prediction of total population size (on the sampled plots):

```
# *Expected* population size for the sample points
getN <- function(fm, newdata=NULL)
  sum(predict(fm, type="lambda", newdata=newdata)[,1])

getN(nb.C2E.C)
[1] 889.6142

# This does the same thing as the following three commands
X <- model.matrix(~chaparral+I(chaparral^2)+elevation+log(offset(area)),
        siteCovs(jayumf))
head(X) # The design matrix
  (Intercept)  chaparral I(chaparral^2)   elevation log(offset(area))
1           1 -0.1218243     0.01484117 -1.20607849          3.341954
2           1  0.8384709     0.70303345 -0.36132054          3.341954
3           1  2.1319298     4.54512461 -0.45797193          3.341954
4           1 -0.2737078     0.07491594 -0.14196112          3.341954
5           1 -1.1562240     1.33685389 -0.72588125          3.341954
6           1  0.2989173     0.08935153  0.01704522          3.341954

# Prediction of total expected population size at the sample points
sum(exp(X %*% c(coef(nb.C2E.C, type="lambda"), 1)))
[1] 889.6142
```

Response curve (chapparal)



Population size estimation

```
attributes(sc.s) # means are "scaled:center". SDs are "scaled:scale"
$dim
[1] 307   4

$dimnames[[2]]
[1] "elevation" "forest"    "chaparral" "area"

$`scaled:center`
  elevation      forest   chaparral        area
202.0023616   0.0673357   0.2703592  28.2743339

$`scaled:scale`
  elevation      forest   chaparral        area
124.8818069   0.1368199   0.2338295   0.0000000
```

Standardize the landscape covariates the same way as the DATA

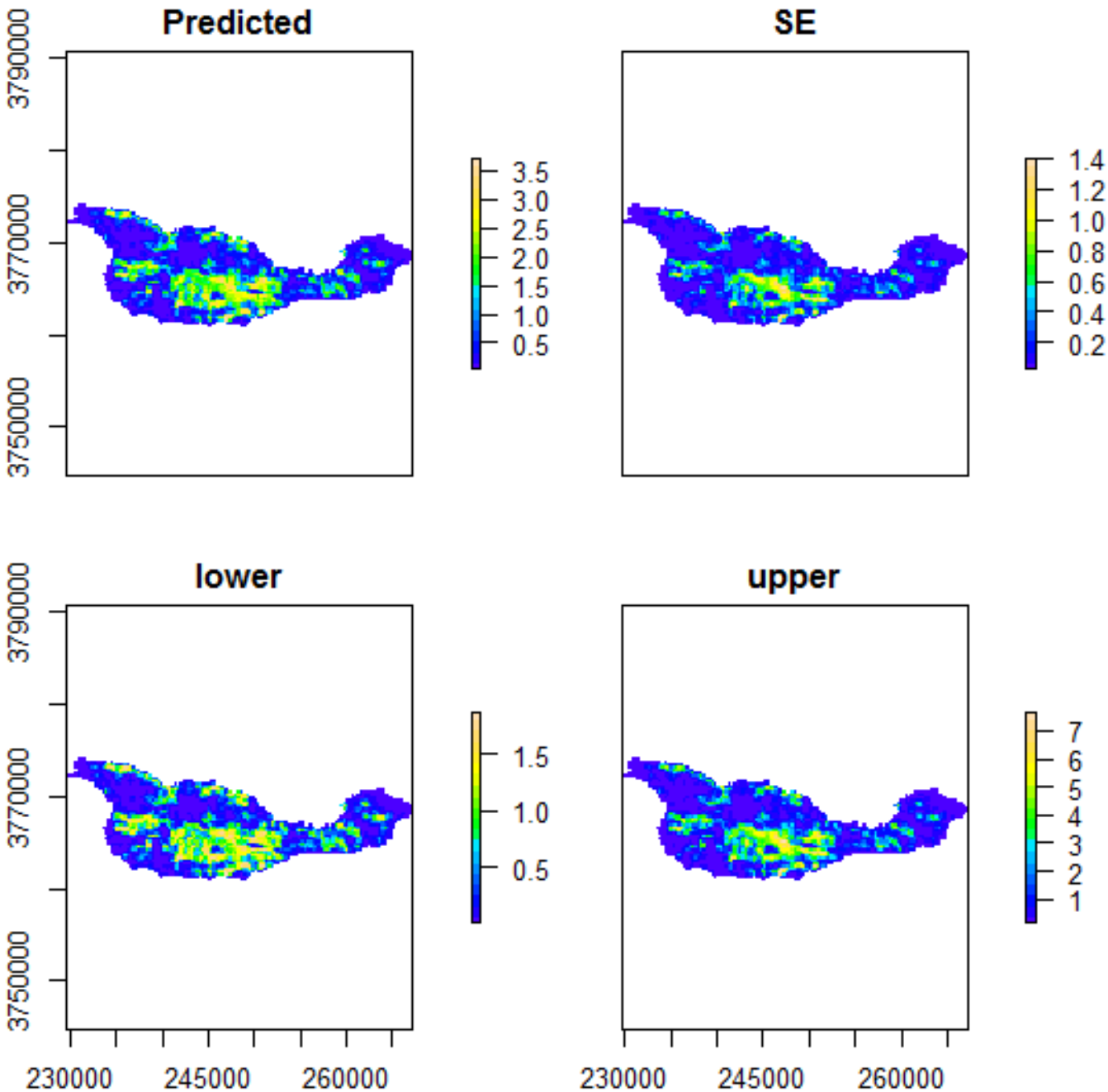
```
cruz.s <- cruz # Created a new data set for the scaled variables
cruz.s$elevation <- (cruz$elevation*0.3048-202)/125   # CHANGED
cruz.s$chaparral <- (cruz$chaparral-0.270)/0.234
cruz.s$area <- (300*300)/10000 # The grid cells are 300x300m=9ha
```

Global population size estimation

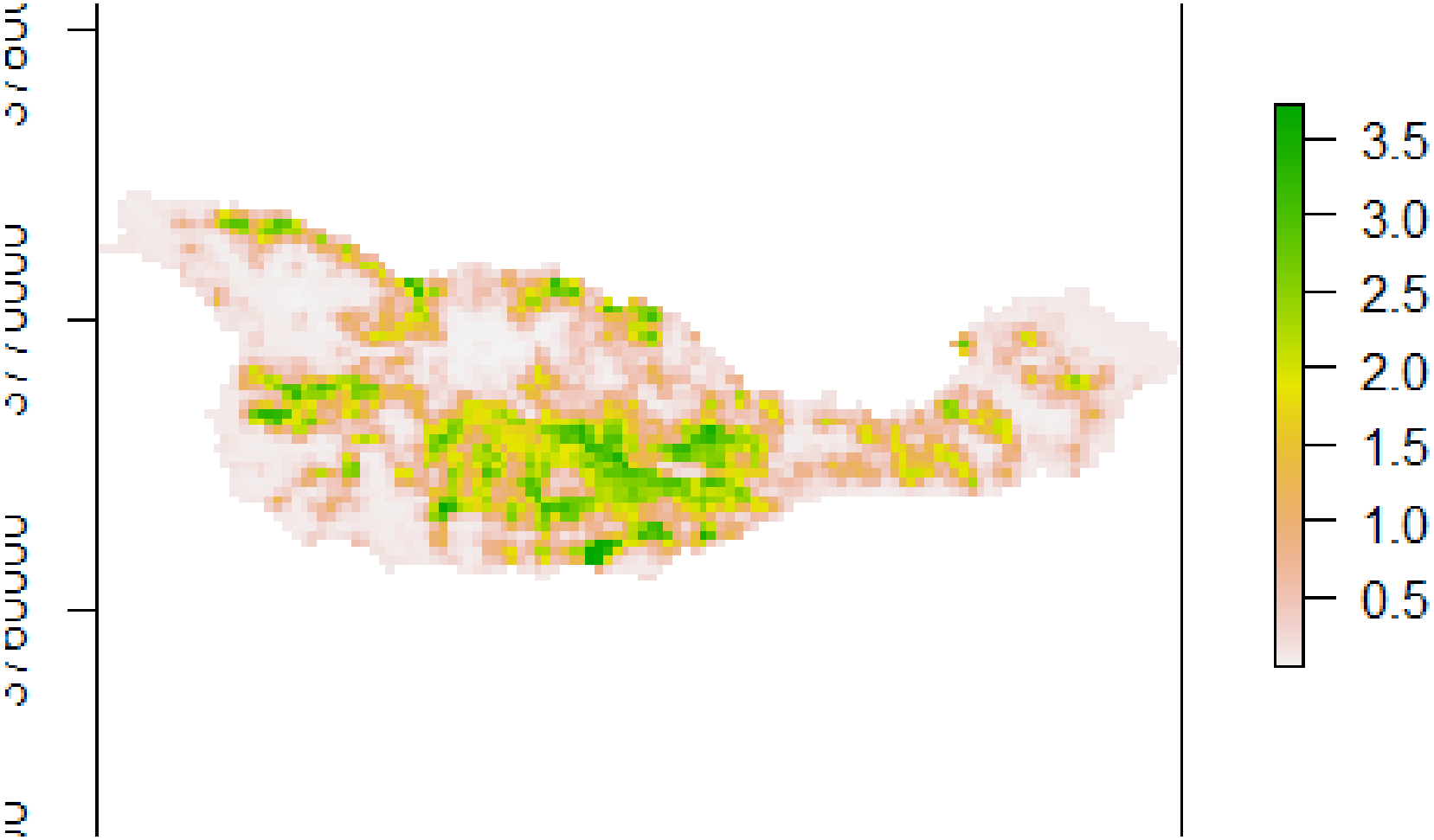
```
# Total population size (by summing predictions for all pixels)

getN(nb.C2E.C, newdata=cruz.s)
  2282.039

# Parametric bootstrap for CI
# A much faster function could be written to doing the sum

(EN.B <- parboot(nb.C2E.C, stat=getN, newdata=cruz.s, nsim=100, report=5))


Call:
parboot(object = nb.C2E.C, statistic = getN, nsim = 100, report = 5, newdata = cruz.s)

Parametric Bootstrap Statistics:
    t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
1 2282            1.1              353        0.495

t_B quantiles:
      0% 2.5%  25%  50%  75% 97.5% 100%
t*1 1450 1666 2010 2284 2526  2930 3310

t0 = Original statistic compuated from data
t_B = Vector of bootstrap samples
```

Mapping

```
library(raster)
cruz.raster <- stack(rasterFromXYZ(cruz.s[,c("x","y","elevation")]),
rasterFromXYZ(cruz.s[,c("x","y","chaparral")]),
rasterFromXYZ(cruz.s[,c("x","y","area")]))

names(cruz.raster) # These should match the names in the formula

plot(cruz.raster) # not shown
# Elevation map on the original scale (not shown).
#    Note: meters here, data in feet.
plot(cruz.raster[["elevation"]]*125 + 202, col=topo.colors(20),
main="Elevation (in meters) and Survey Locations", asp = 1)
points(issj[,c("x","y")], cex=0.8, pch = 16)

EN <- predict(nb.C2E.C, type="lambda", newdata=cruz.raster)
plot(EN , col = topo.colors(20), asp = 1) # See Figure 8.8
```


Predicted
SE
lower
upper
3790000
3770000
3750000
230000
245000
260000
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1.4
1.2
1.0
0.8
0.6
0.4
0.2
1.5
1.0
0.5
7
6
5
4
3
2
1

```
plot(EN[["Predicted"]])
```



Summary

- HDS addresses one of the fundamental problems in ecological studies: modeling spatial variation in population size or density
- `unmarked` has unique capabilities for fitting HDS models which can be found in no other software
- Next: Bayesian analysis of HDS models