Appendix S1

Matthew T. Farr, David S. Green, Kay E. Holekamp, Gary J. Roloff, and Elise F. Zipkin

Multi-species hierarchical modeling reveals variable responses of African carnivores to management alternatives

Ecological Applications

Simulation of winding survey bias

Straight line survey routes were infeasible due to impassible terrain and off-road restrictions; thus surveys were designed to maximize coverage of the Talek region and Mara Triangle (Figure 1).

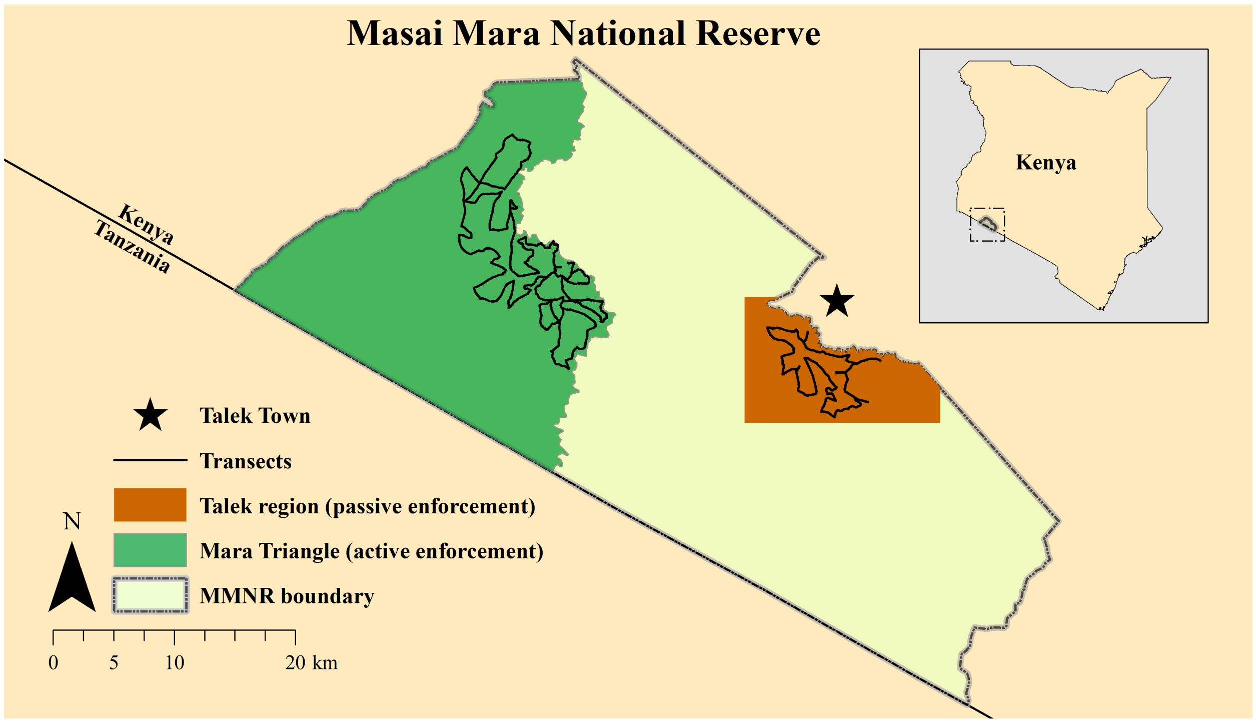
Masai Mara National Reserve
Kenya
Kenya
Tanzania
Talek Town
Transects
Talek region (passive enforcement)
Mara Triangle (active enforcement)
MMNR boundary
N
0
5
10
20 km

Figure S1. Map of survey routes in the Talek and Mara Triangle management regions within the Masai Mara National Reserve, Kenya. Transect vehicle surveys were conducted at 4 to 6 week intervals from July 2012 to March 2014.

The curvatures of winding surveys do not violate the assumption of distance sampling or pose a serious issue to estimating detection probabilities (Hiby & Krishna 2001). However, winding surveys may violate the assumption of random placement, which ensures a representative sample of distances to observations for estimation of detection probability (Buckland et al. 1993). We conducted a simulation study to examine any biases created by winding surveys and to compare performance to straight surveys with random placement. We checked for biases in the scale parameter, σ, for detection probability, the number of groups within (i.e., Groups within) the sampling boundary (i.e., transect width of 1300 m), the abundance (derived from mean number of groups and mean group size) within the sampling boundary (i.e., Abundance within), the number of groups within the entire study region (i.e., Groups), and the abundance within the entire study region (i.e., Abundance).

To do this, we generate count data for a single species across the study region by simulating the location of groups assuming a uniform distribution for the intensity of groups. We then simulate group sizes for each group following the model in the main text. We sample from the simulated data using distance sampling with both winding and staight survey designs. We then compare the relative biases of the estimated parameters and the true parameters between winding and straight survey designs using a hierarchical distance sampling model (i.e., does not include a multi-species component).

We generated 100 datasets for simulating the locations of groups and group sizes and resampled each sampling design 10 times for a total of 1000 simulations for each sampling design. For each dataset, observations (groups not individuals) were distributed uniformly across the sampling area. Group sizes were then simulated for each group. Then each dataset of uniformly distributed observations was resampled 10 times.

The results of the simulation study show that the winding survey is slightly more biased (8.87 % more biased) for the number of groups within the sampling boundary (i.e., Groups within) when compared to the straight survey (Table 1).

Table S1. The mean relative biases (percent) for our 5 parameters of interest for the winding survey design and the straigh survey design when compared to the true values.

```
             Winding Straight
Groups In      17.43     8.36
Abundance In   11.22    12.95
Groups         14.09     7.65
Abundance      10.54     9.35
Sigma           5.29    14.33
```

This also holds true (winding surveys were 6.44% more biased) for the number of groups within the enitre study region (i.e., Groups). However, abundance within the sampling boundary (i.e., Abundance within) and for the entire study region (i.e., Abundance) were similar between the two survey methods, and there was no increase in bias for the scale parameter, σ, which influenced detection. We concluded that the winding survey design did not have a signifcant enough increase in bias to impact our results. Additionally, we assume that any potential biases caused by winding transects would have a similar influence in both management regions and would only affect absolute, but not relative, abundance estimates (summary of results, including the remaining tables and figures, are presented after the annotated code).

Below is the annotated code for the simulation study for a single simulation. To see complete code for the simulation study, please go to GitHub.

Annotated code

Set seed

```
set.seed(1985)
```

Load R packages

```
library(rgdal)
library(sp)
library(dplyr)
library(tidyr)
library(jagsUI)
```

Create study region UTM boundaries where individuals will be simulated

```r
#Easting UTM
xlim <- c(715304, 752393)

#Northing UTM
ylim <- c(9831970, 9857296)
```

Simulate true latent values

```r
#Number of groups
N <- 1000

#Simulate UTM coordinates of groups
u1 <- runif(N, xlim[1], xlim[2])
u2 <- runif(N, ylim[1], ylim[2])

#Group size
lambda.group <- 2
cs <- rpois(N, lambda.group) + 1

#Abundance
Ntotal <- sum(cs)

#Half normal scale parameter
sigma <- 300

#Mid point of each distance class
midpt <- seq(12.5, 650, 25)

#Index for distance class
nG <- length(midpt)

#Width of distance class
v <- 25

#Transect half width
B <- 650
```

Create winding sampling design

```r
#Directory for sampling design shapefiles
d.dir <- "./Transects"

#Import transect shapefiles
Site1 <- readOGR(dsn = d.dir, layer = "Site1")
Site2 <- readOGR(dsn = d.dir, layer = "Site2")
Site3 <- readOGR(dsn = d.dir, layer = "Site3")
Site4 <- readOGR(dsn = d.dir, layer = "Site4")
Site5 <- readOGR(dsn = d.dir, layer = "Site5")
Site6 <- readOGR(dsn = d.dir, layer = "Site6")
Site7 <- readOGR(dsn = d.dir, layer = "Site7")
Site8 <- readOGR(dsn = d.dir, layer = "Site8")
```

```
Site9 <- readOGR(dsn = d.dir, layer = "Site9")
Site10 <- readOGR(dsn = d.dir, layer = "Site10")
Site11 <- readOGR(dsn = d.dir, layer = "Site11")
Site12 <- readOGR(dsn = d.dir, layer = "Site12")
Site13 <- readOGR(dsn = d.dir, layer = "Site13")
Site14 <- readOGR(dsn = d.dir, layer = "Site14")
Site15 <- readOGR(dsn = d.dir, layer = "Site15")
Site16 <- readOGR(dsn = d.dir, layer = "Site16")
Site17 <- readOGR(dsn = d.dir, layer = "Site17")
```

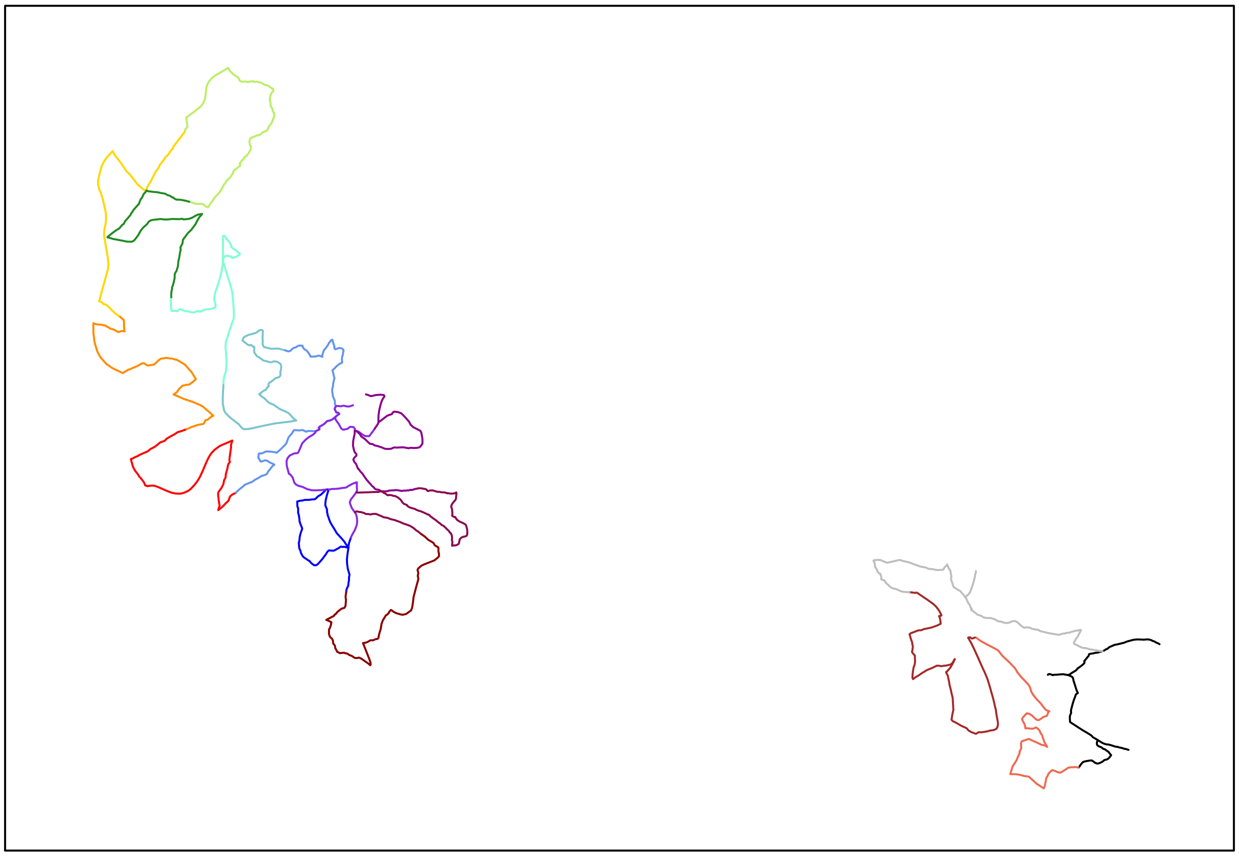

Figure S2. Imported transect shapefiles for the winding survey. Each of the 17 transects is represented by a different color.

Sample coordintaes from transect. Used to calculate distances of observed groups.

```
s1p <- spsample(Site1, 30, type = "regular")
s2p <- spsample(Site2, 30, type = "regular")
s3p <- spsample(Site3, 30, type = "regular")
s4p <- spsample(Site4, 30, type = "regular")
s5p <- spsample(Site5, 30, type = "regular")
s6p <- spsample(Site6, 30, type = "regular")
s7p <- spsample(Site7, 30, type = "regular")
s8p <- spsample(Site8, 30, type = "regular")
s9p <- spsample(Site9, 30, type = "regular")
s10p <- spsample(Site10, 30, type = "regular")
s11p <- spsample(Site11, 30, type = "regular")
s12p <- spsample(Site12, 30, type = "regular")
```

```r
s13p <- spsample(Site13, 30, type = "regular")
s14p <- spsample(Site14, 30, type = "regular")
s15p <- spsample(Site15, 30, type = "regular")
s16p <- spsample(Site16, 30, type = "regular")
s17p <- spsample(Site17, 30, type = "regular")
```

Combine site coordinates

```r
#Easting
X <- c(s1p@coords[,1], s2p@coords[,1], s3p@coords[,1], s4p@coords[,1],
       s5p@coords[,1], s6p@coords[,1], s7p@coords[,1], s8p@coords[,1],
       s9p@coords[,1], s10p@coords[,1], s11p@coords[,1], s12p@coords[,1],
       s13p@coords[,1], s14p@coords[,1], s15p@coords[,1], s16p@coords[,1],
       s17p@coords[,1])

#Northing
Y <- c(s1p@coords[,2], s2p@coords[,2], s3p@coords[,2], s4p@coords[,2],
       s5p@coords[,2], s6p@coords[,2], s7p@coords[,2], s8p@coords[,2],
       s9p@coords[,2], s10p@coords[,2], s11p@coords[,2], s12p@coords[,2],
       s13p@coords[,2], s14p@coords[,2], s15p@coords[,2], s16p@coords[,2],
       s17p@coords[,2])
```

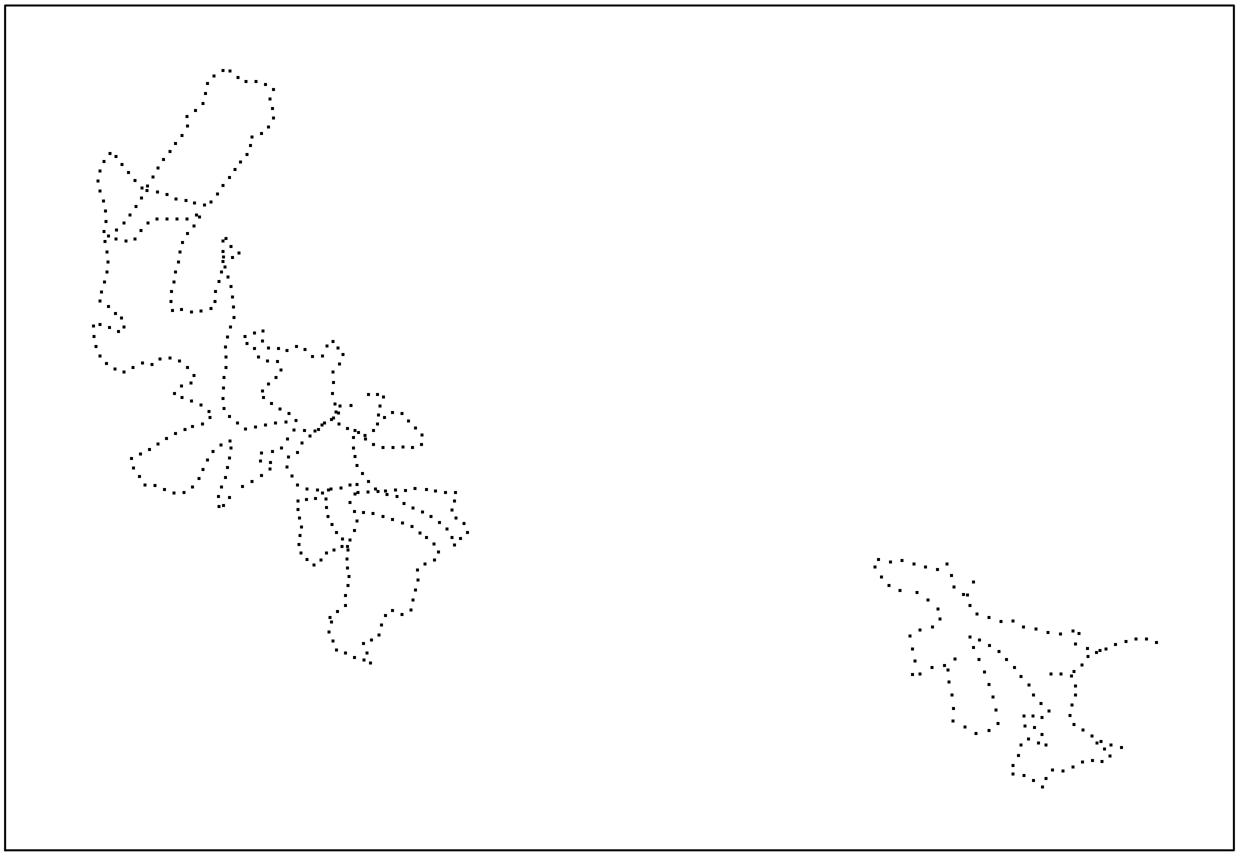

Figure S3. Breaking continous winding survey into discrete points for calculation of distance from observation to transect.

Initialize values

```r
#Index for sites
nsites <- 17

#Index for transect points
J <- length(X)

#ID for sites
si <- seq(0, J, (J/nsites))

#ID for distance class
di <- seq(0,650,25)

#Minimum distance value
dst <- rep(NA, N)

#ID for nearest site
q <- rep(NA, N)

#Site
site <- rep(NA, N)

#Distance value to each transect point
d <- array(NA, dim = c(N, J))

#ID for groups less than 650 meters
y <- rep(NA, N)

#Index recorder
index <- rep(NA, N)
```

Simulate distances and site of groups

```r
for(i in 1:N){
  for(j in 1:J){

    #Distance from each group to each point on the transect
    d[i,j] <- sqrt((u1[i] - X[j])^2 + (u2[i] - Y[j])^2)
  }

  #Distance to nearest point on the transect
  dst[i] <- min(d[i,])

  #Index of which point in 1:J is the nearest
  q[i] <- which.min(d[i,])

  for(j in 1:nsites){

    #Determine the site for each group
    if(si[j] < q[i] && q[i] <= si[j+1])
      site[i] <- j
  }

  #Index of which observation are within 650 meters of transect
  if(dst[i] < 650)
    y[i] <- 1
  index[i] <- i
}
```

Harvest simulated data

```r
#Dataframe that includes information on all groups
Dtot <- cbind(y, index, u1, u2, site, cs)

#Dataframe containing only groups within 650 meters to transect
Din <- Dtot[complete.cases(Dtot),]

#Number of groups within 650 meters
Nin <- length(Din[,1])

#Abundance within 650 meters
Nintotal <- sum(Din[,6])
```

Initialize data

```r
#Remove groups not within 650 meters
index <- index[y==1]
index <- index[!is.na(index)]

#Detection Probability
p <- NULL

#Number of captured ("detected") groups
ncap <- rep(NA, Nin)

#Distance Class
dclass <- rep(NA, Nin)
```

Simulate detection of groups less than 650 meters

```r
for(i in 1:Nin){

  #Detection probability using half-normal distance function
  p[i] <- exp(-dst[index[i]] * dst[index[i]] / (2 * sigma * sigma))

  #Simulate number of groups detected
  ncap[i] <- rbinom(1, 1, p[i])

  for(k in 1:nG){

    #Determine distance class for each group
    if(di[k] < dst[index[i]] && dst[index[i]] <= di[k+1])
      dclass[i] <- k
  }
}
```

Harvest simulated data

```r
#Add distance class, detection probability, and detection index to dataframe
Din <- cbind(Din[,2:6],dclass, p, ncap)

#Undetected groups as NAs
for(i in 1:Nin){
  if(Din[i,8] == 0)
    Din[i,8] <- NA
}

#Dataframe of detected inidividuals
Dcap <- Din[complete.cases(Din),]

#Create observed number of groups per site
y.new <- table(Dcap[,4])
y.new <- as.data.frame(y.new)
colnames(y.new) <- c("site", "freq")
y.new$site <- as.integer(y.new$site)
y.new <- tbl_df(y.new)

#Add in sites with no detections
miss <- y.new %>% expand(site = 1:nsites)
miss$freq <- rep(0, length(miss))

#Add missing sites into observed groups per site
yobs <- full_join(y.new, miss, by = "site")
yobs <- yobs %>% arrange(site)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```r
yobs <- as.numeric(yobs$freq.x)
yobs[is.na(yobs)] <- 0

#Site index for observed number of groups
site <- Dcap[,4]

#Distance class index for observed number of groups
dclass <- Dcap[,6]

#Number of observations
nobs <- sum(yobs)

#Group size
gs <- Dcap[,5]
```

Create offset for sites with longer transects and sampling area

```r
#Search area (meters squared) of each site
A.site <- as.vector(c(11.6542, 11.9619, 12.4702, 12.5182, 10.7843, 10.2384, 10.7495,
                      12.0545, 9.0114, 11.2589, 10.4075, 9.7834, 11.8226, 10.5295,
                      11.5376, 14.8511, 14.0352))
```

JAGS Model

```
cat("
    model{

    ##Priors

    for(j in 1:nsites){

    #Abundance prior
    alpha[j] ~ dnorm(0, 0.01)

    #Detection prior
    sigma[j] ~ dunif(0, 500)

    }#End j loop

    #OVerdispersion prior
    r.N ~ dunif(0,100)
    r.G ~ dunif(0,100)

    #Group size prior
    beta ~ dunif(0, 50)

    ##Likelihood

    #Multinomial detection component
    for(i in 1:nobs){

    dclass[i] ~ dcat(fc[1:nG, site[i]])

    }#End i loop

    for(j in 1:nsites){

    #Construct cell probabilities for nG cells
    for(k in 1:nG){

    #Half normal detection function at midpt (length of rectangle)
    p[k,j] <- exp(- midpt[k] * midpt[k] / (2 * sigma[j] * sigma[j]))

    #Probability of x in each interval (width of rectangle)
    pi[k,j] <- v/B

    #Detection probability for each interval (area of each rectangle)
    f[k,j] <- p[k,j] * pi[k,j]

    #Conditional detection probability (scale to 1)
    fc[k,j] <- f[k,j] / pcap[j]

    }#End k loop

    #Detection probability at each site (sum of rectangles)
    pcap[j] <- sum(f[1:nG,j])
    #Observation process
    y[j] ~ dbin(pcap[j], N[j])

    #Description of latent number of groups (negative binomial)
    N[j] ~ dpois(lambda.star[j])

    #Expected Number of Groups
    lambda.star[j] <- rho[j] * lambda[j]

    #Overdispersion parameter for Expected Number of Groups
    rho[j] ~ dgamma(r.N, r.N)

    #Linear model for number of groups
    lambda[j] <- exp(alpha[j] + log(offset[j]))

    #Expected Group Size
    gs.lam.star[j] <- gs.lam[j] * gs.rho[j]

    #Overdispersion parameter for Expected Group Size
    gs.rho[j] ~ dgamma(r.G, r.G)

    #Group size
    gs.lam[j] <- exp(beta)

    }#End j loop

    for(i in 1:nobs){

    gs[i] ~ dpois(gs.lam.star[site[i]]) T(1,)

    }#End i loop

    ##Derived quantities

    #Number of groups within sampling boundary
    Nin <- sum(N[1:nsites])

    for(j in 1:nsites){

    #Abundance at each transect
    Ntotal[j] <- lambda.star[j] * gs.lam.star[j]

    } #End j loop

    #Abundance within sampling boundary
    Nintotal <- sum(Ntotal[])

    #Proportion of study region covered by sampling design
    D <- (939.316/164.4837)

    #Number of groups in entire study region
    Nwinding <- Nin * D
```

```
    #Abundance in entire study region
    Nwindingtotal <- Nintotal * D

    }",fill=TRUE, file="HMSDS_model.txt")
```

Compile JAGS data

```
#Input data
str(windingD <- list(nG = nG, v = v, site = site, y = yobs, B = B, midpt = midpt,
                 nobs = nobs, dclass = dclass, nsites = nsites,
                 gs = gs, offset = A.site))
```

```
## List of 11
##  $ nG    : int 26
##  $ v     : num 25
##  $ site  : num [1:97] 2 6 3 4 1 14 14 4 9 7 ...
##  $ y     : num [1:17] 5 5 6 9 1 3 4 7 6 4 ...
##  $ B     : num 650
##  $ midpt : num [1:26] 12.5 37.5 62.5 87.5 112.5 ...
##  $ nobs  : num 97
##  $ dclass: num [1:97] 9 9 11 22 13 7 11 18 6 2 ...
##  $ nsites: num 17
##  $ gs    : num [1:97] 2 1 2 2 2 3 2 2 2 4 ...
##  $ offset: num [1:17] 11.7 12 12.5 12.5 10.8 ...
```

```
#Initial values
N.in <- yobs + 1

inits <- function(){list(N = N.in, sigma = runif(17, 50, 350))}

#Parameters to monitor
params<-c('sigma', 'Nin', 'Nintotal', 'Nwinding', 'Nwindingtotal')

#MCMC settings

nc <- 3
ni <- 12000
nb <- 2000
nt <- 4
```

Run model

```
windingM <- jags(data = windingD, model.file = "HMSDS_model.txt",
                 inits = inits, parameters.to.save = params,
             n.chains = nc, n.iter = ni, n.burnin = nb, n.thin = nt)
```

Save and remove data for next sampling

```
windingVals <- list(cbind(Din[,2], Din[,3]), cbind(Dcap[,2], Dcap[,3]), Nin, Nintotal)

rm(X, Y, nsites, J, si, di, dclass, dst, q,
   site, d, y, index, Dtot, Din, Nin, Nintotal,
   p, ncap, Dcap, y.new, miss, yobs, nobs, gs,
   N.in, inits)
```

Create straight sampling design. There are 10 transects that run north to south.

```r
#Sampling area middle UTM coordinate
mdE <- 733848.5
mdN <- 9844633

#Sampling area left corner UTM coordinate
Et <- mdE - (13000/2)
Nt <- mdN + (12650/2)

#Sample points from straight transects
Ep <- seq((Et + 650), (Et + (13000 - 650)), 1300)
Np <- seq(Nt, (Nt - 12650), -253)
X <- rep(Ep, rep(length(Np), length(Ep)))
Y <- rep(Np, length(Ep))
```

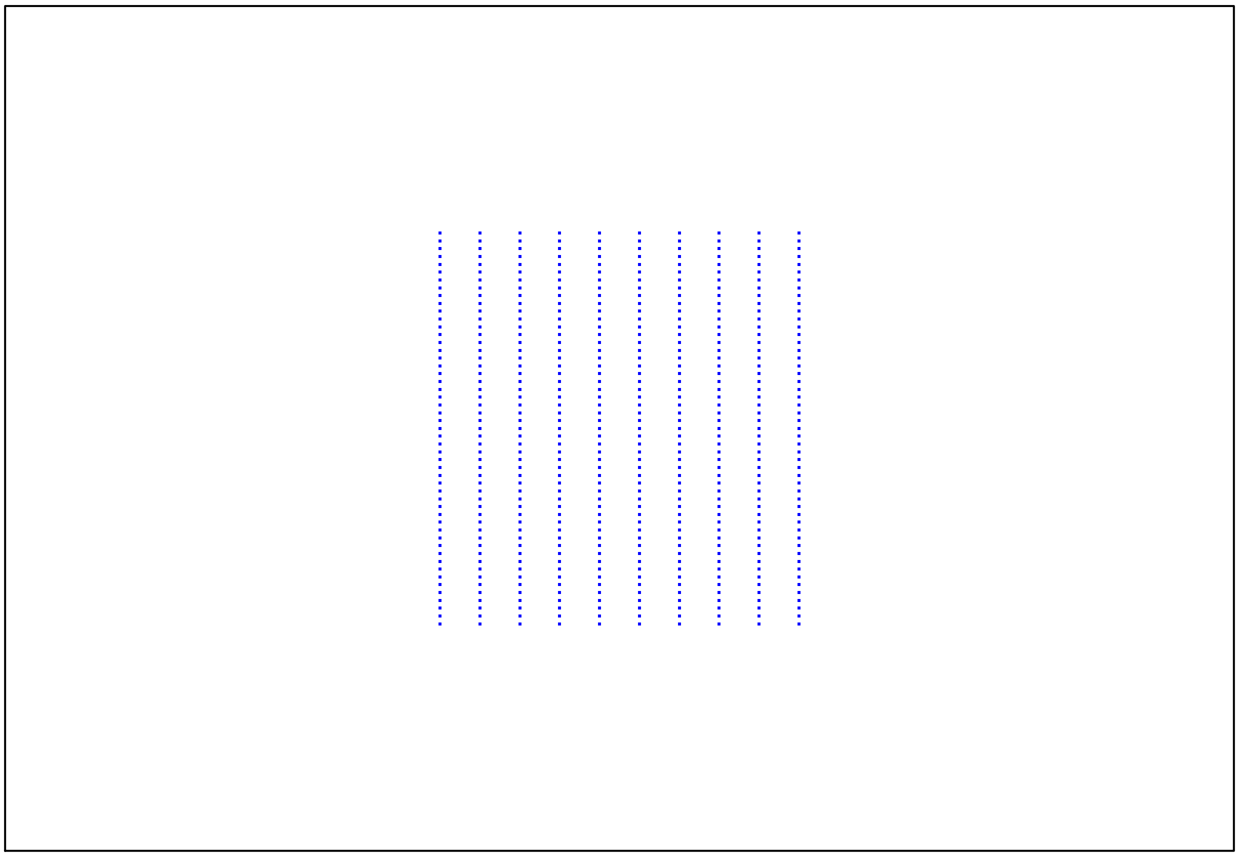

Figure S4. Transects for straight survey design that have been discretized into points for distance sampling calculations.

Initialize values

```r
#Index for sites
nsites <- 10

#Index for transect points
J <- length(X)
```

```r
#ID for sites
si <- seq(0, J, (J/nsites))

#ID for distance class
di <- seq(0,650,25)

#Minimum distance value
dst <- rep(NA, N)

#ID for nearest site
q <- rep(NA, N)

#Site
site <- rep(NA, N)

#Distance value to each transect point
d <- array(NA, dim = c(N, J))

#ID for groups less than 650 meters
y <- rep(NA, N)

#Index recorder
index <- rep(NA, N)
```

Simulate data for distances and site for groups

```r
for(i in 1:N){
  for(j in 1:J){

    #Distance from each group to each point on the transect
    d[i,j] <- sqrt((u1[i] - X[j])^2 + (u2[i] - Y[j])^2)
  }

  #Distance to nearest point on the transect
  dst[i] <- min(d[i,])

  #Index of which point in 1:J is the nearest
  q[i] <- which.min(d[i,])

  for(j in 1:nsites){

    #Determine the site for each group
    if(si[j] < q[i] && q[i] <= si[j+1])
      site[i] <- j
  }

  #Index of which observation are within 650 meters of transect
  if(dst[i] < 650)
    y[i] <- 1
  index[i] <- i
}
```

Harvest simulated data

```r
#Dataframe that includes information on all groups
Dtot <- cbind(y, index, u1, u2, site, cs)

#Dataframe containing only groups within 650 meters to transect
Din <- Dtot[complete.cases(Dtot),]

#Number of groups within 650 meters
Nin <- length(Din[,1])

#Abundance within 650 meters
Nintotal <- sum(Din[,6])
```

Initialize data

```r
#Remove groups not within 650 meters
index <- index[y==1]
index <- index[!is.na(index)]

#Detection Probability
p <- NULL

#Number of captured ("detected") groups
ncap <- rep(NA, Nin)

#Distance Class
dclass <- rep(NA, Nin)
```

Simulate detection of groups less than 650 meters

```r
for(i in 1:Nin){

  #Detection probability using half-normal distance function
  p[i] <- exp(-dst[index[i]] * dst[index[i]] / (2 * sigma * sigma))

  #Simulate number of groups detected
  ncap[i] <- rbinom(1, 1, p[i])

  for(k in 1:nG){

    #Determine distance class for each group
    if(di[k] < dst[index[i]] && dst[index[i]] <= di[k+1])
      dclass[i] <- k
  }
}
```

Harvest simulated data

```r
#Add distance class, detection probability, and detection index to dataframe
Din <- cbind(Din[,2:6],dclass, p, ncap)

#Undetected groups as NAs
for(i in 1:Nin){
  if(Din[i,8] == 0)
    Din[i,8] <- NA
}

#Dataframe of detected inidividuals
Dcap <- Din[complete.cases(Din),]

#Create observed number of groups per site
y.new <- table(Dcap[,4])
y.new <- as.data.frame(y.new)
colnames(y.new) <- c("site", "freq")
y.new$site <- as.integer(y.new$site)
y.new <- tbl_df(y.new)

#Add in sites with no detections
miss <- y.new %>% expand(site = 1:nsites)
miss$freq <- rep(0, length(miss))

#Add missing sites into observed groups per site
yobs <- full_join(y.new, miss, by = "site")
yobs <- yobs %>% arrange(site)
yobs <- as.numeric(yobs$freq.x)
yobs[is.na(yobs)] <- 0

#Site index for observed number of groups
site <- Dcap[,4]

#Distance class index for observed number of groups
dclass <- Dcap[,6]

#Number of observations
nobs <- sum(yobs)

#Group size
gs <- Dcap[,5]
```

Compile JAGS data. Reuse JAGS model, parameters to save, and MCMC settings.

```
#Input data
str(altD <- list(nG = nG, v = v, site = site, y = yobs, B = B, midpt = midpt,
                 nobs = nobs, dclass = dclass, nsites = nsites,
                 gs = gs, offset = rep(1, nsites)))
```

```
## List of 11
##  $ nG    : int 26
##  $ v     : num 25
##  $ site  : num [1:94] 10 2 8 10 1 2 8 2 3 7 ...
##  $ y     : num [1:10] 14 14 11 8 4 6 5 12 8 12
##  $ B     : num 650
##  $ midpt : num [1:26] 12.5 37.5 62.5 87.5 112.5 ...
##  $ nobs  : num 94
##  $ dclass: num [1:94] 10 5 1 7 19 7 21 1 11 13 ...
##  $ nsites: num 10
##  $ gs    : num [1:94] 1 4 3 2 2 1 4 3 8 2 ...
##  $ offset: num [1:10] 1 1 1 1 1 1 1 1 1 1
```

```
#Initial values
N.in <- yobs + 1

inits <- function(){list(N = N.in, sigma = runif(10, 50, 350))}
```

Run JAGS model

```
altM <- jags(data = altD, model.file = "HMSDS_model.txt",
             inits = inits, parameters.to.save = params,
             n.chains = nc, n.iter = ni, n.burnin = nb, n.thin = nt)
```

Save and remove data

```
altVals <- list(cbind(Din[,2], Din[,3]), cbind(Dcap[,2], Dcap[,3]), Nin, Nintotal,
                cbind(X, Y))

rm(X, Y, nsites, J, si, di, dclass, dst, q,
   site, d, y, index, Dtot, Din, Nin, Nintotal,
   p, ncap, Dcap, y.new, miss, yobs, nobs, gs,
   N.in, inits)
```

Absoulte relative bias estimates

```
bias <- t(matrix(data = c(

#Number of groups in search area

#Winding True
windingVals[[3]],

#Winding Estimate
windingM$mean$Nin,

#Winding Bias
(abs(mean((windingM$sims.list$Nin - windingVals[[3]])/windingVals[[3]])) * 100),

#Straight True
altVals[[3]],

#Straight Estimate
altM$mean$Nin,

#Straight Bias
(abs(mean((altM$sims.list$Nin - altVals[[3]])/altVals[[3]])) * 100),

#Abundance in search area

#Winding True
windingVals[[4]],
```

```r
#Winding Estimate
windingM$mean$Nintotal,

#Winding Bias
(abs(mean((windingM$sims.list$Nintotal - windingVals[[4]])/windingVals[[4]])) * 100),

#Straight True
altVals[[4]],

#Straight Estimate
altM$mean$Nintotal,

#Straight Bias
(abs(mean((altM$sims.list$Nintotal - altVals[[4]])/altVals[[4]])) * 100),

#Number of groups in survey boundary

#Winding True
N,

#Winding Estimate
windingM$mean$Nwinding,

#Winding Bias
(abs(mean((windingM$sims.list$Nwinding - N)/N)) * 100),

#Straight True
N,

#Straight Estimate
altM$mean$Nwinding,

#Straight Bias
(abs(mean((altM$sims.list$Nwinding - N)/N)) * 100),

#Abundance in survey boundary

#Winding True
Ntotal,

#Winding Estimate
windingM$mean$Nwindingtotal,

#Winding Bias
(abs(mean((windingM$sims.list$Nwindingtotal - Ntotal)/Ntotal)) * 100),

#Straight True
Ntotal,

#Straight Estimate
altM$mean$Nwindingtotal,

#Straight Bias
```

```r
(abs(mean((altM$sims.list$Nwindingtotal - Ntotal)/Ntotal)) * 100),

#Scale parameter

#Winding True
sigma,

#Winding Estimate
mean(windingM$mean$sigma),

#Winding Bias
(abs(mean((rowMeans(windingM$sims.list$sigma) - sigma)/sigma)) * 100),

#Straight True
sigma,

#Straight Estimate
mean(altM$mean$sigma),

#Straight Bias
(abs(mean((rowMeans(altM$sims.list$sigma) - sigma)/sigma)) * 100)),

nrow = 6, ncol = 5))

colnames(bias) <- c("Winding True", "Winding Est", "Winding Bias",
                    "Straight True", "Straight Est", "Straight Bias" )
rownames(bias) <- c("Groups Within", "Abundance Within",
                    "Groups", "Abundance", "Sigma")
```

Results

Table S2. Estimates from winding survey model.

```
                mean         sd       2.5%      25%      50%      75%    97.5%      Rhat
sigma[1]    361.5381  82.725309  205.35510 296.8184 365.2291 432.6505 492.0257 1.0002786
sigma[2]    351.7562  86.113652  192.04970 284.0781 353.4002 424.8410 491.8415 1.0035031
sigma[3]    329.7888  88.143471  181.31174 258.5118 323.2260 399.3612 489.6031 0.9998238
sigma[4]    381.0933  72.079791  240.61825 326.1833 385.1793 442.5452 493.9590 1.0005661
sigma[5]    284.8812 116.229939   96.28678 186.5329 276.2936 382.7636 488.0244 0.9998437
sigma[6]    299.4937 100.998976  135.44460 216.2327 289.6298 380.5779 487.3339 1.0021628
sigma[7]    272.4059  98.247177  127.16693 193.2126 253.9217 344.2314 480.5698 1.0009632
sigma[8]    323.9865  86.990206  182.96313 252.7636 314.2280 392.3209 487.8310 0.9998602
sigma[9]    366.8639  79.920184  213.34166 303.9001 370.9093 434.8347 493.6132 0.9999023
sigma[10]   278.0095  97.873709  134.08969 198.9767 259.7507 348.0000 479.7637 1.0008481
sigma[11]   241.9426 125.129424   61.60525 134.5135 221.5143 340.1916 482.4633 1.0006223
sigma[12]   245.8121  83.416387  135.35979 185.2865 225.3152 288.0898 459.0312 1.0010212
sigma[13]   268.9899  79.225063  158.69776 210.4126 251.5630 311.6830 464.4278 1.0011127
sigma[14]   380.4420  75.949032  227.48984 322.7248 387.5950 444.1131 494.5186 1.0009136
sigma[15]   288.4602  85.748004  161.44227 221.5431 270.9757 345.7137 478.4204 1.0015018
sigma[16]   270.3612  99.969423  124.65026 190.1569 250.7451 340.9392 482.4233 1.0005929
sigma[17]   269.7516  76.658582  163.82445 212.4646 253.3669 312.1305 458.8460 1.0002058
Nin         189.8449  19.407330  156.00000 176.0000 188.5000 202.0000 231.0000 1.0000438
```

```
Nintotal        488.6801  71.130586  363.39545  438.4800  483.9685  533.7685  640.2641 0.9999589
Nwinding       1084.1462 110.829311  890.86819 1005.0821 1076.4657 1153.5601 1319.1702 1.0000438
Nwindingtotal 2790.7024 406.204978 2075.24002 2504.0248 2763.7960 3048.1881 3656.3523 0.9999589
deviance        967.2747   6.738697  955.70330  962.4959  966.6961  971.5198  982.0931 1.0004535
```

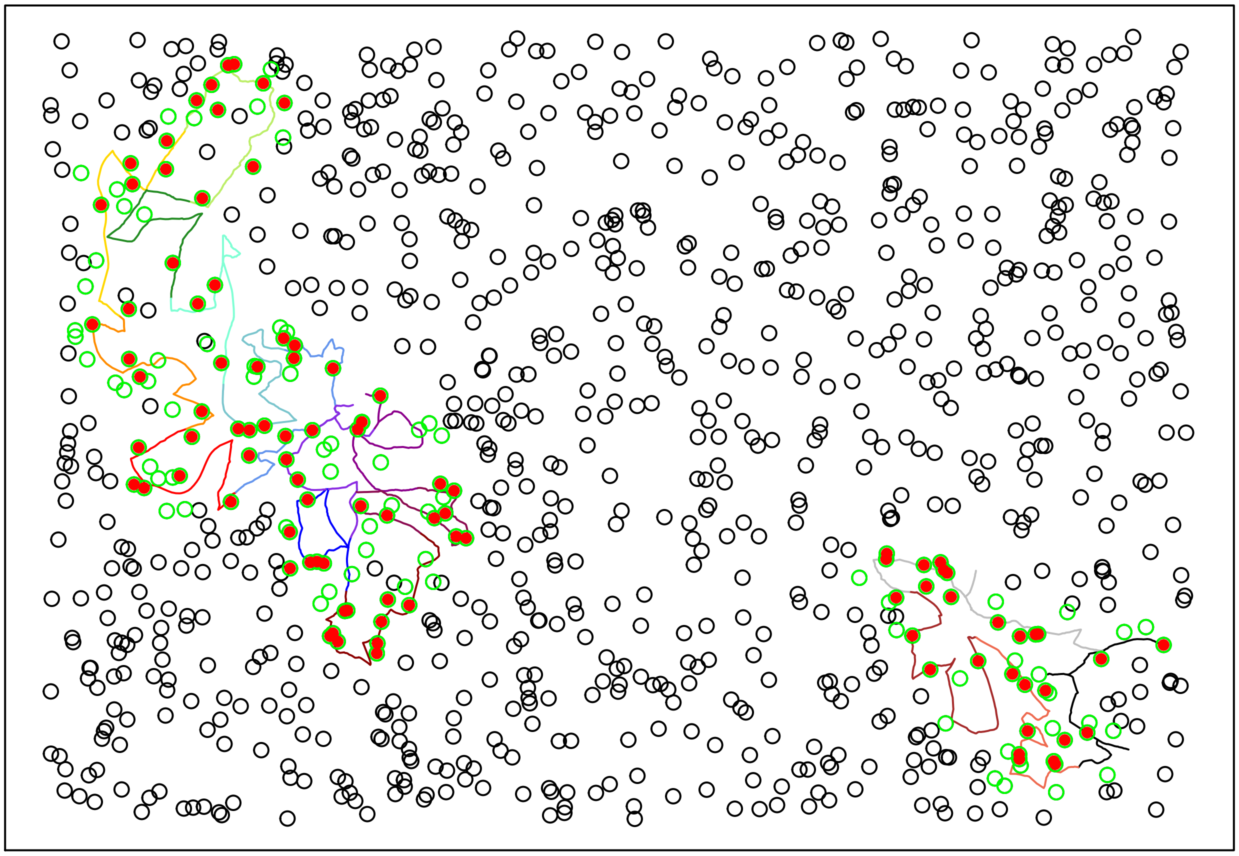

Figure S5. Visualization of winding survey. Black circles indicate simulated observations not within sampling boundary. Green circles with no fill indicate observations within sampling boundary that were not detected. Green circles with red fill indicate observations within sampling boundary that were detected.

Table S3. Estimates from straight survey model.

```
                   mean         sd       2.5%       25%       50%       75%     97.5%     Rhat
sigma[1]       356.2561  74.541733  228.1617  296.5602  353.1144  416.4099  489.9215 1.000923
sigma[2]       390.6489  67.585823  258.0860  338.5735  395.3510  447.4029  495.0198 1.000074
sigma[3]       416.4657  58.613220  287.7742  376.8880  426.3411  465.2430  496.5902 1.000051
sigma[4]       245.6722  82.143796  138.0335  184.9791  226.2021  286.6003  454.7032 1.000682
sigma[5]       208.8332  95.488361   92.6865  137.1835  182.0562  257.1388  453.7360 1.000066
sigma[6]       297.3220  92.293125  155.3468  222.7820  281.7395  367.7493  480.5558 1.001138
sigma[7]       355.6178  84.271770  201.2798  287.7379  357.3418  427.7083  491.7101 1.001657
sigma[8]       370.1056  72.292024  236.1250  313.4340  369.8965  429.8846  492.2168 1.002394
sigma[9]       362.8274  79.594719  216.6240  299.9492  363.9505  430.7197  492.4527 1.000542
sigma[10]      227.3926  65.633465  142.3310  182.0690  212.2046  255.4479  409.6643 1.000407
Nin            172.6891  18.095960  142.0000  160.0000  171.0000  184.0000  211.0000 1.002670
Nintotal       478.6867  70.311607  356.3549  429.2852  473.9663  522.7498  634.0965 1.001564
Nwinding       986.1743 103.340484  810.9185  913.7110  976.5286 1050.7676 1204.9563 1.002670
Nwindingtotal 2733.6332 401.528041 2035.0339 2451.5164 2706.6766 2985.2640 3621.1305 1.001564
deviance       933.4153   6.486467  922.2968  928.8231  932.9405  937.4109  947.3177 1.008891
```

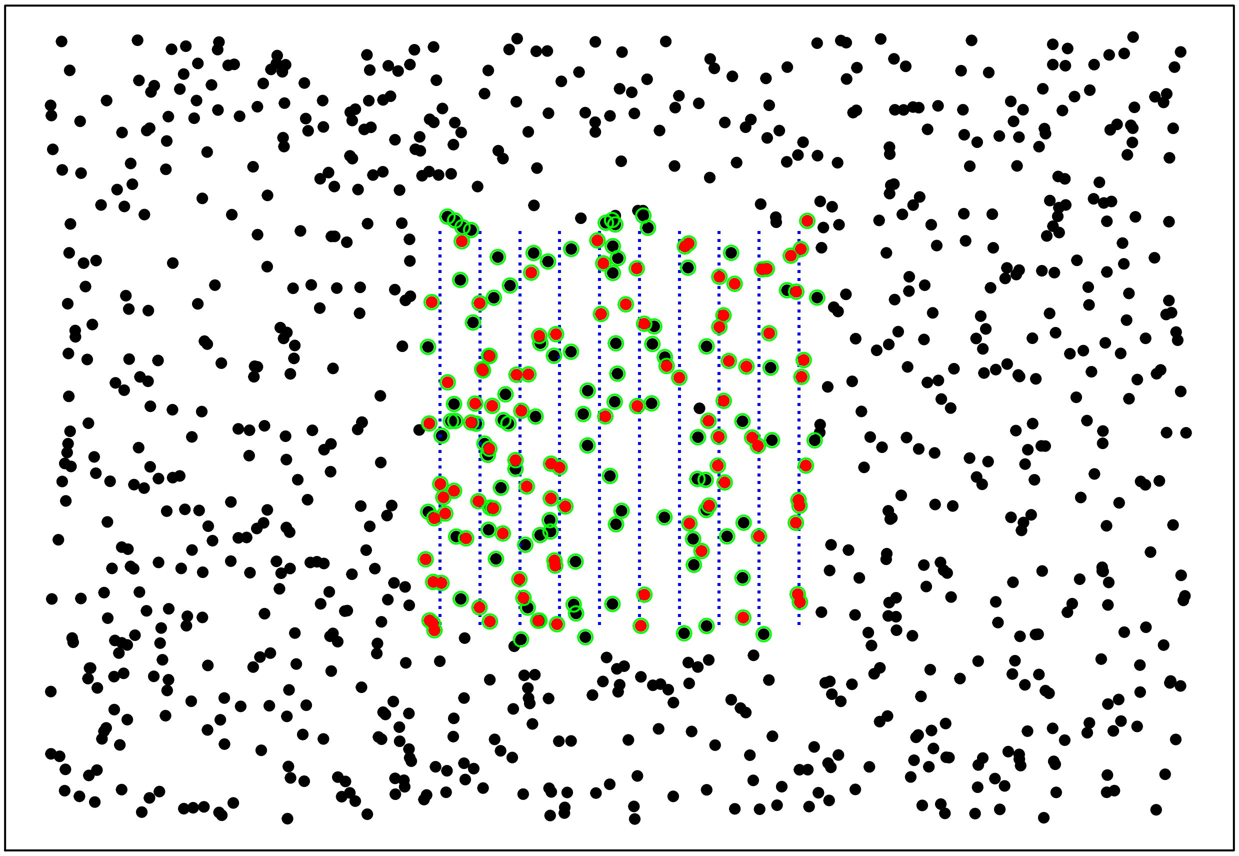

Figure S6. Visualization of straight survey. Black circles indicate simulated observations not within sampling boundary. Green circles with no fill indicate observations within sampling boundary that were not detected. Green circles with red fill indicate observations within sampling boundary that were detected.

Table S4. Estimated and true values with associated relative bias for winding and straight surveys for a single dataset.

	Winding True	Winding Est	Winding Bias	Straight True	Straight Est	Straight Bias
Groups Within	168	189.8449	13.002937	185	172.6891	6.654559
Abundance Within	477	488.6801	2.448664	535	478.6867	10.525846
Groups	1000	1084.1462	8.414623	1000	986.1743	1.382567
Abundance	2960	2790.7024	5.719512	2960	2733.6332	7.647528
Sigma	300	306.7986	2.266209	300	323.1141	7.704716

Literature Cited

Buckland, S.T., Anderson, D.R., Burnham, K.P. & Laake, J.L. (1993) Distance Sampling: Estimating Abundance of Biological Populations. Oxford University Press, Oxford.

Hiby, L. & Krishna, M.B. (2001) Line transect sampling from a curving path. Biometrics, 57, 727-731.