Supporting Information

Appendix A: 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).

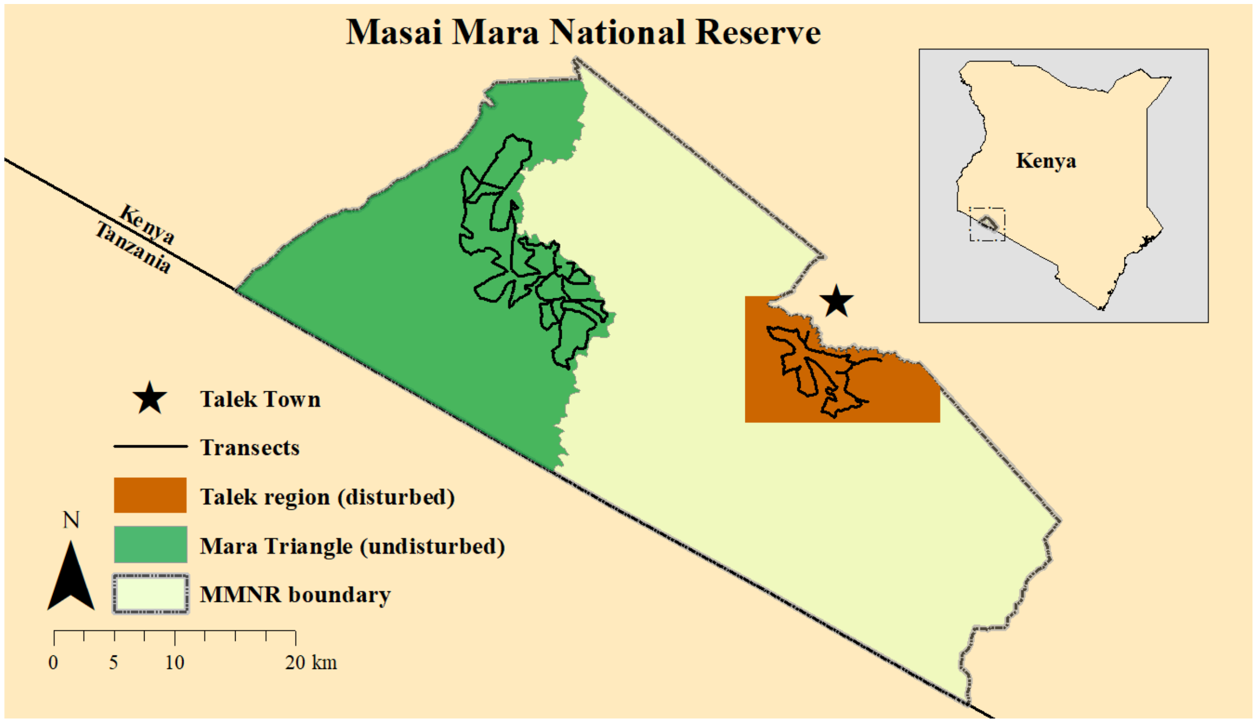

Figure 1. Map of survey routes in the Talek and Mara Triangle management regions within the Masai Mara National Reserve, Kenya. Transect vehicle surveys were conducted at 4 to 6-week intervals from July 2012 to March 2014.

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 1. 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

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

Load R packages

```r
library(rgdal)
```

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

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

```r
library(sp)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```r
library(tidyr)
```

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

```r
library(jagsUI)
```

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

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")
```

```r
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")
```

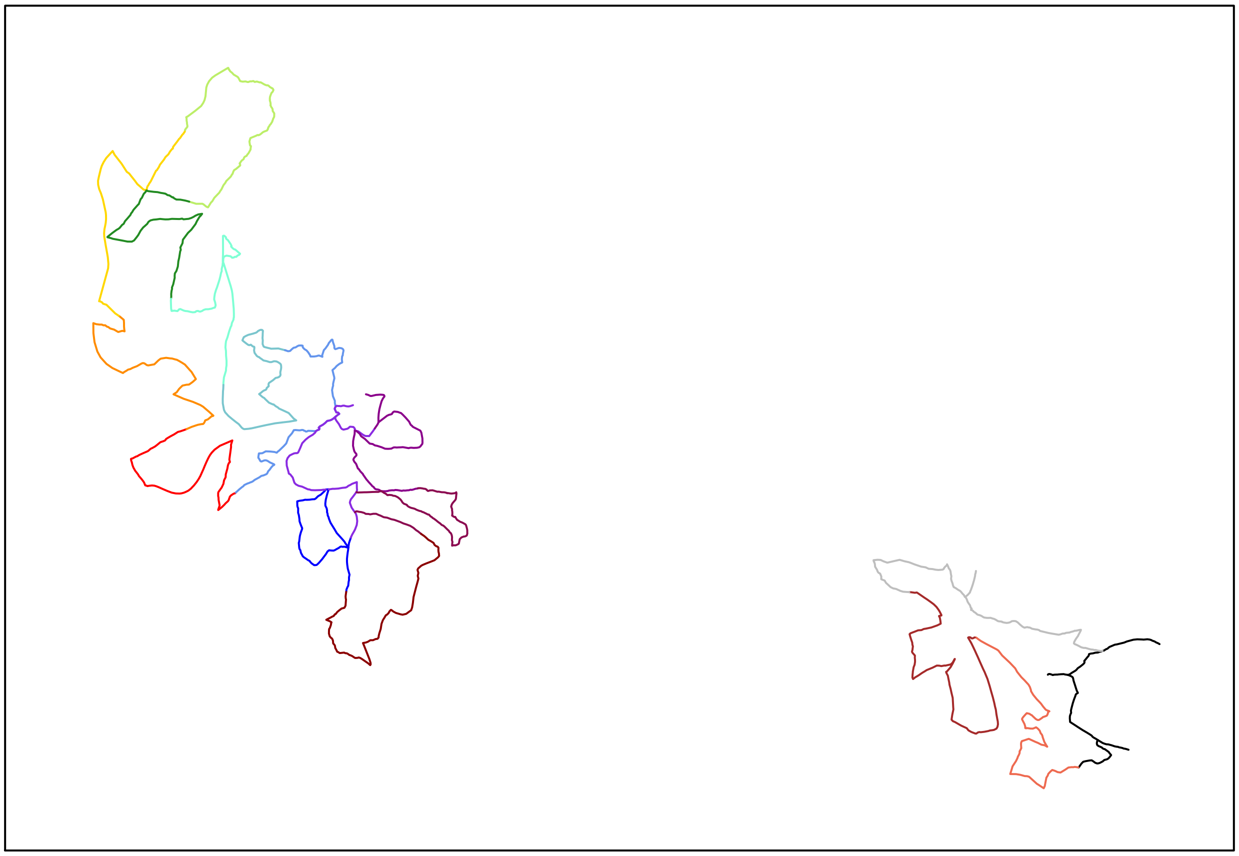

Figure 2. Imported transect shapefiles for the winding survey. Each of the 17 transects is represented by a different color.

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

```r
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")
```

```r
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")
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])
```

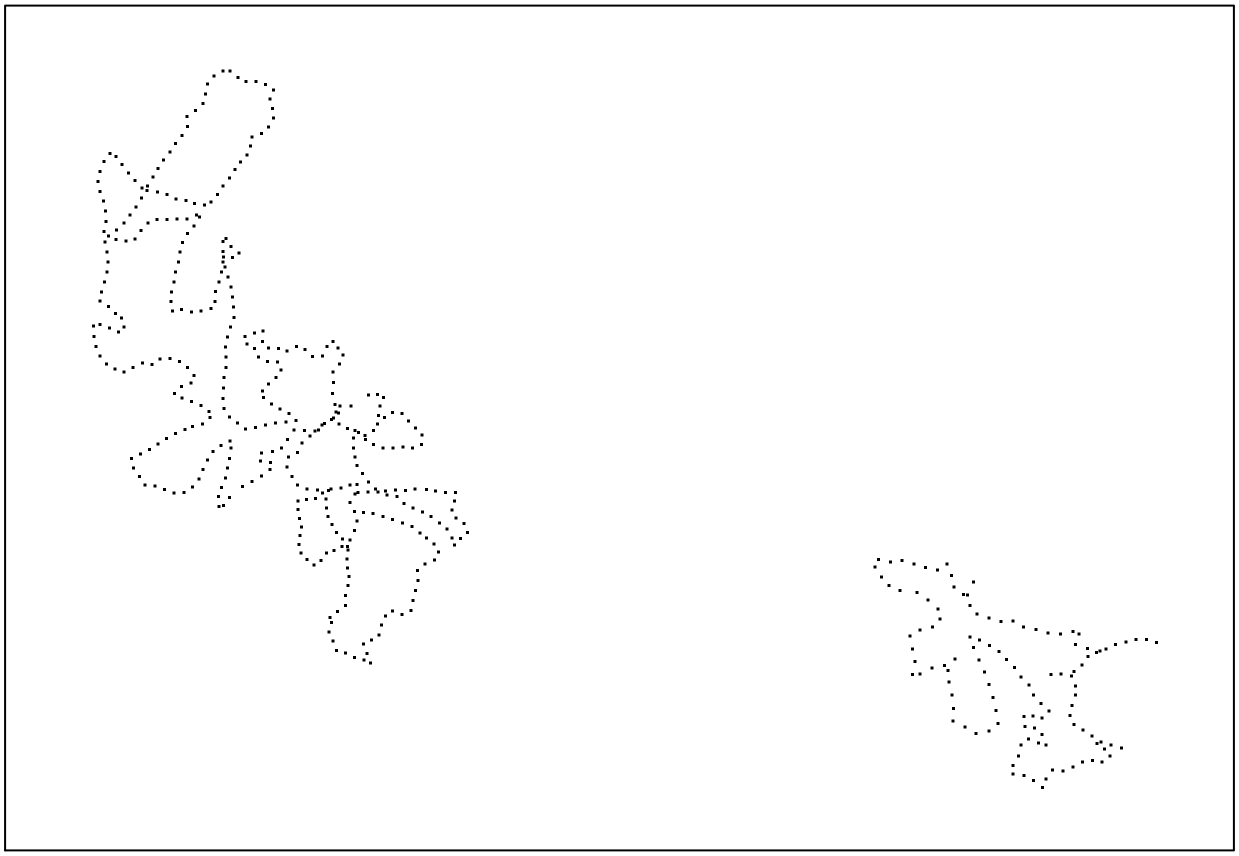

Figure 3. Breaking continous winding survey into discrete points for calculation of distance from observation to transect.

Initialize values

```
#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
```

```r
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
```

```r
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]
```

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))
```

BUGS Model

```r
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 BUGS 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 ...
```

```r
#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

```r
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)
```

```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 211
##    Unobserved stochastic nodes: 88
##    Total graph size: 3500
##
## Initializing model
##
## Adaptive phase.....
## Adaptive phase complete
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 10000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

Save and remove data for next sampling

```r
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))
```

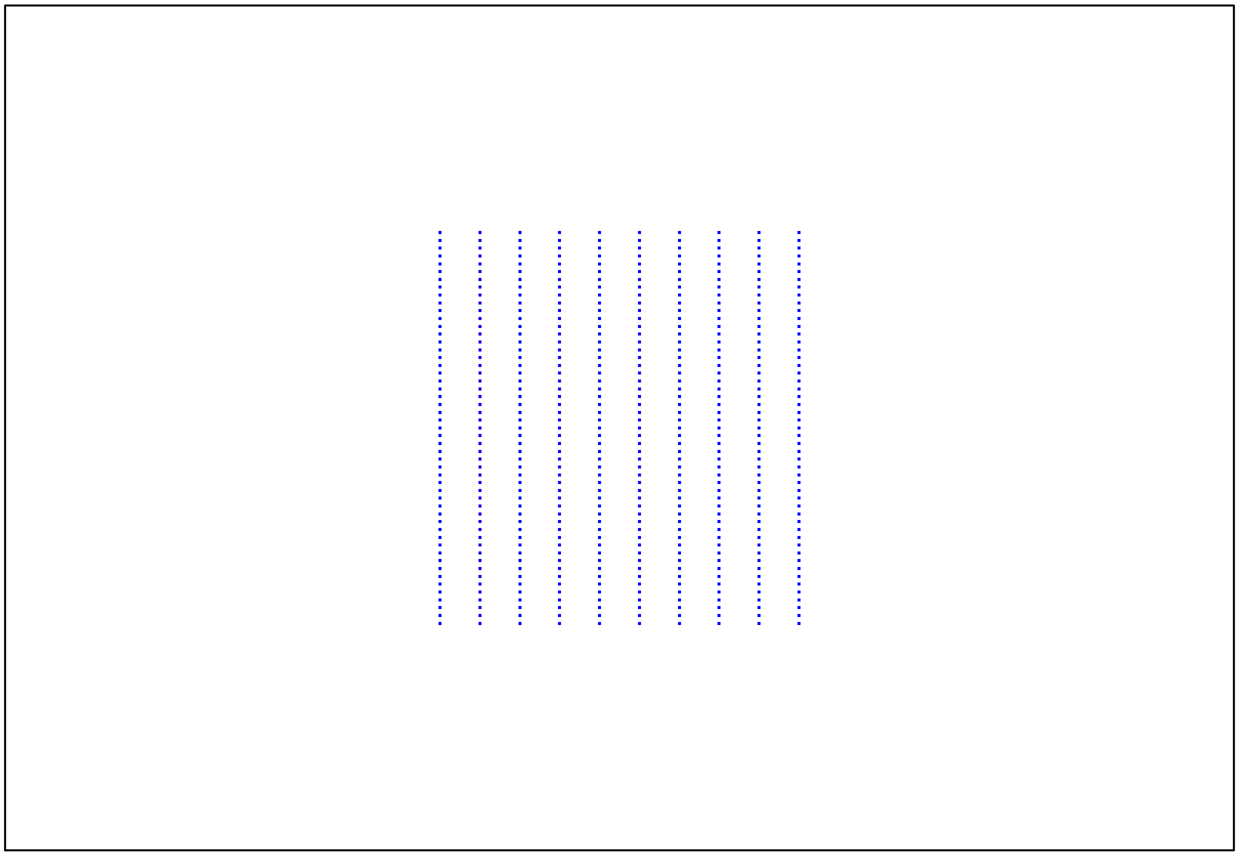

Figure 4. Transects for straight survey design that have been discretized into points for distance sampling calculations.

Initialize values

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

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

#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

```
#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

```
#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

```
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 BUGS data. Reuse BUGS model, parameters to save, and MCMC settings.

```r
#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 BUGS 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)
```

```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 198
##    Unobserved stochastic nodes: 53
##    Total graph size: 2240
##
## Initializing model
##
## Adaptive phase.....
## Adaptive phase complete
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 10000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

Save and remove data

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

```r
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

```r
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]],

#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
(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
```

```r
(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 2. Estimates from winding survey model.

```
                   mean        sd       2.5%       25%       50%       75%     97.5%      Rhat
sigma[1]       360.1094  81.90293  206.16143  296.8129  361.3790  428.9270  492.2782 1.0003847
sigma[2]       350.0135  85.36746  195.23623  281.8588  350.0495  421.5482  491.3344 0.9999762
sigma[3]       329.7722  88.53780  181.68803  257.8640  324.3811  402.6156  488.1728 1.0009326
sigma[4]       382.4086  72.14885  241.77323  326.6358  388.2601  443.8110  493.9026 1.0000557
sigma[5]       283.5815 116.23717   95.91588  185.3856  274.6006  381.4619  488.7421 1.0000574
sigma[6]       298.5340 101.36226  134.02791  214.3548  290.4844  379.3527  486.8146 1.0021552
sigma[7]       273.7872  99.58882  127.22162  194.9066  255.4191  344.2816  481.7115 1.0006154
sigma[8]       326.2450  87.37319  181.56607  255.6213  318.7279  397.0497  486.5710 1.0002973
sigma[9]       365.9549  79.32776  216.96898  303.8309  369.1218  433.9260  492.3054 1.0000844
sigma[10]      281.6216  97.97003  132.85660  203.1910  264.8248  353.5207  481.4707 1.0003675
sigma[11]      244.1141 125.87565   60.47840  136.1830  222.3787  344.1518  484.1709 1.0011430
sigma[12]      240.0745  79.51743  135.09585  181.8824  221.3633  277.7420  449.4790 1.0039531
sigma[13]      270.3981  79.86169  158.50305  209.5329  253.3835  315.8004  461.9023 1.0007597
sigma[14]      385.4948  75.03341  230.90766  330.0711  394.6808  448.9463  494.7625 1.0013452
sigma[15]      291.4802  87.35146  163.40233  222.4014  274.2164  350.5709  480.1483 1.0002308
sigma[16]      272.6532 100.35674  125.86940  191.7249  252.8479  346.2357  480.3270 1.0008134
sigma[17]      268.8899  76.25258  164.08852  212.6584  251.6024  309.5062  459.4519 1.0002899
Nin            190.4433  20.39466  155.00000  176.0000  189.0000  203.0000  234.0000 1.0019879
Nintotal       489.7460  72.63816  363.75231  437.8542  485.1127  534.4652  646.1972 1.0009548
Nwinding      1087.5635 116.46767  885.15750 1005.0821 1079.3211 1159.2708 1336.3023 1.0019879
Nwindingtotal 2796.7891 414.81429 2077.27798 2500.4510 2770.3299 3052.1668 3690.2342 1.0009548
deviance       967.2711   6.69438  955.48071  962.5124  966.7953  971.5187  981.7181 0.9998918
```

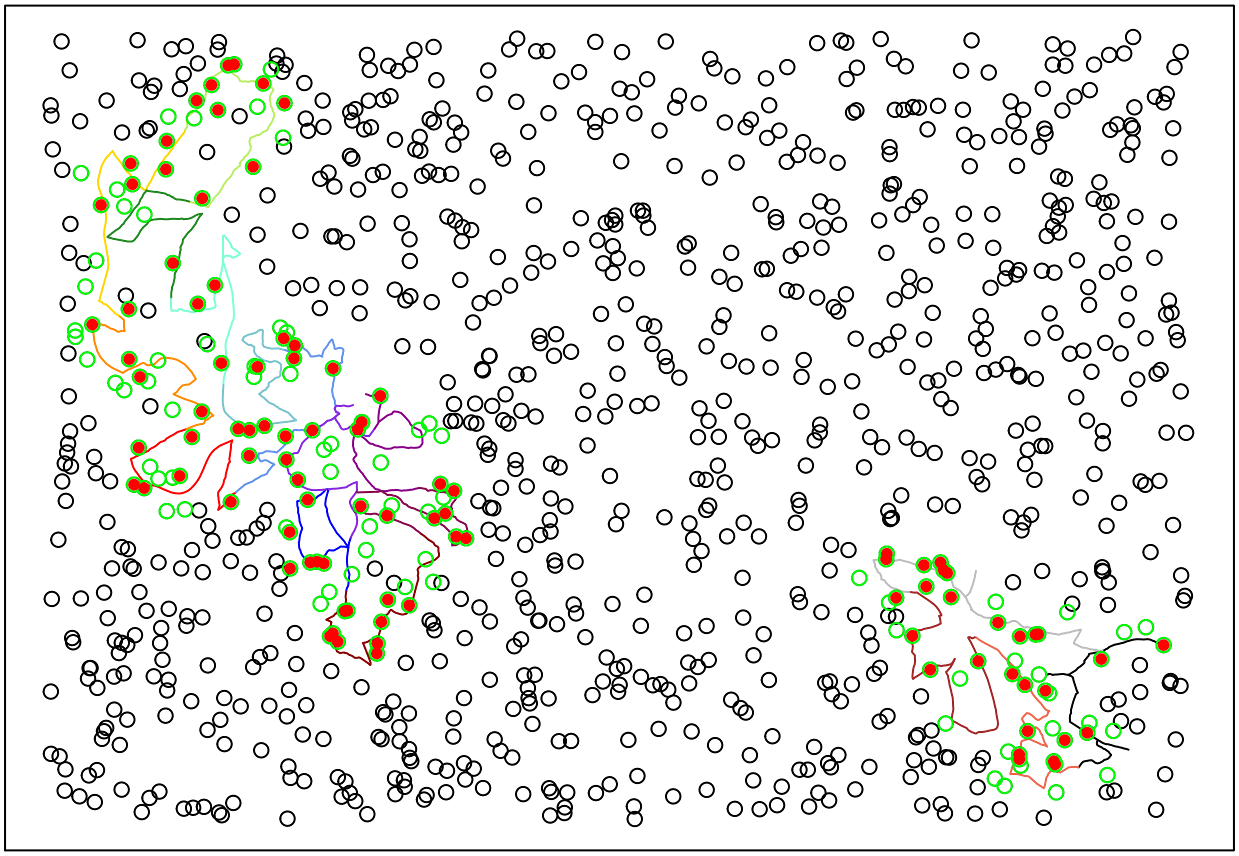

Figure 5. Visualization of winding survey. Black circles indicate simulated observations not within sampling boundary. Green circles with no fill indicate observations within sampling boundary that were not detected. Green circles with red fill indicate observations within sampling boundary that were detected.

Table 3. Estimates from straight survey model.

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat
sigma[1]	354.4664	74.193820	226.99166	295.7218	350.3078	413.2952	489.2051	1.0012311
sigma[2]	387.2176	67.648396	255.88241	335.4203	390.6792	443.7040	494.2562	1.0002469
sigma[3]	414.4723	58.684130	288.53390	373.7434	422.4571	463.7419	496.3955	1.0009826
sigma[4]	251.0232	84.901009	138.57009	187.3557	229.0158	297.2533	460.5826	1.0013272
sigma[5]	202.7813	92.706511	89.28482	133.5188	177.5716	247.5769	446.4410	1.0020797
sigma[6]	294.4422	91.493012	155.10113	221.2081	278.1584	359.7346	481.3641	1.0003050
sigma[7]	356.7427	84.559092	199.77243	289.7537	359.8768	428.8878	492.7097	1.0004908
sigma[8]	369.6672	73.697326	233.59265	310.9419	371.0417	430.2958	492.6363	0.9999558
sigma[9]	360.2184	78.624062	217.75844	298.1276	359.4995	425.4465	492.0885	1.0035682
sigma[10]	228.5749	66.424748	143.15223	182.5637	212.5337	257.5818	412.3781	1.0041760
Nin	173.3039	18.327096	141.00000	161.0000	172.0000	185.0000	213.0000	1.0000643
Nintotal	481.9135	70.982358	356.98009	431.1365	476.8530	527.4263	632.9155	1.0007671
Nwinding	989.6853	104.660429	805.20779	919.4217	982.2393	1056.4783	1216.3777	1.0000643
Nwindingtotal	2752.0606	405.358491	2038.60391	2462.0886	2723.1611	3011.9697	3614.3863	1.0007671
deviance	934.1502	6.274647	923.04127	929.7240	933.6748	938.1643	947.7615	1.0006409

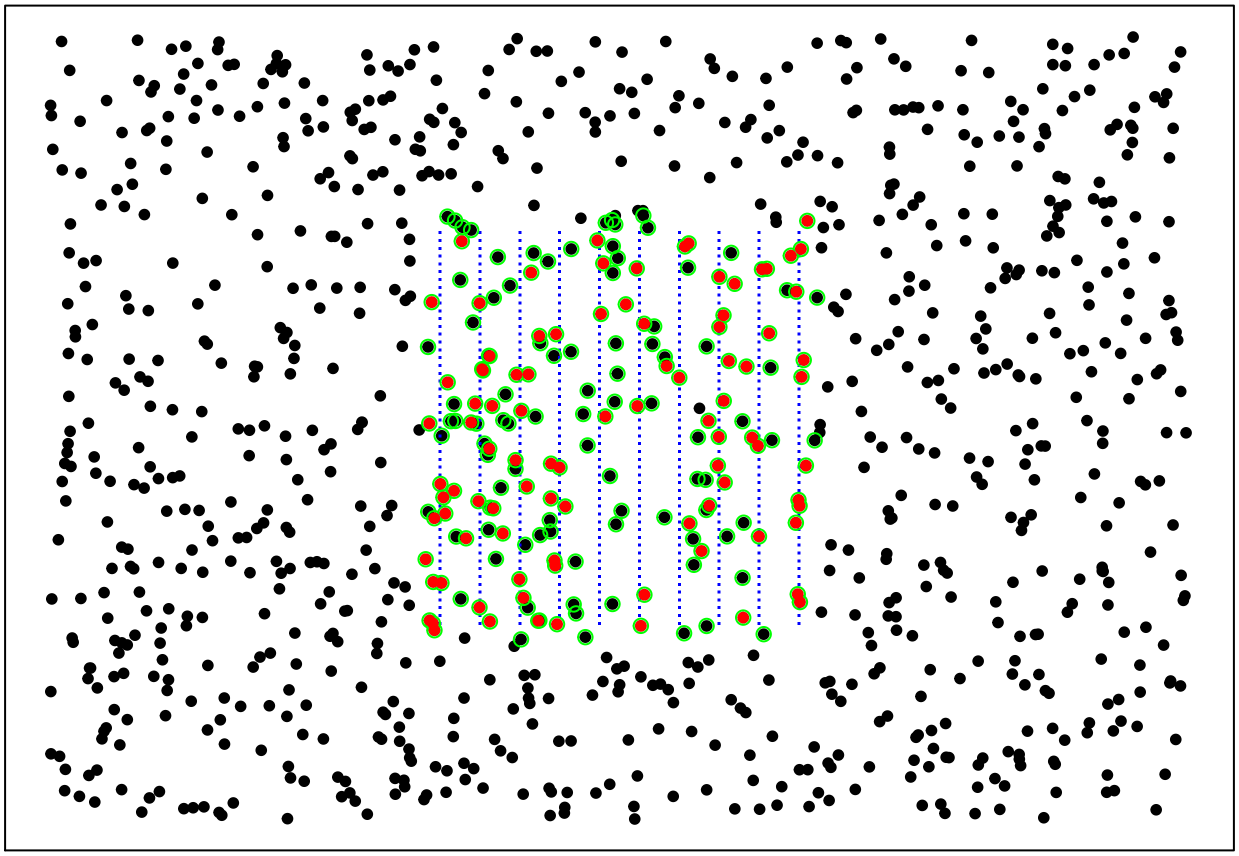

Figure 6. Visualization of straight survey. Black circles indicate simulated observations not within sampling boundary. Green circles with no fill indicate observations within sampling boundary that were not detected. Green circles with red fill indicate observations within sampling boundary that were detected.

Table 4. 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	190.4433	13.359127	185	173.3039	6.322234
Abundance Within	477	489.7460	2.672111	535	481.9135	9.922702
Groups	1000	1087.5635	8.756351	1000	989.6853	1.031473
Abundance	2960	2796.7891	5.513881	2960	2752.0606	7.024981
Sigma	300	307.3607	2.453582	300	321.9606	7.320206

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.

Appendix B: Multi-species distance sampling JAGS code

```
model{

#--------#
#-PRIORS-#
#--------#

#Gamma0
mu_s ~ dunif(0, 8)            #Mean
tau_s <- 1/(sig_s * sig_s)    #Precision
sig_s ~ dunif(0, 8)           #Variance

#Sigma
gamma1 ~ dnorm(0, 0.01)       #Effect of body size
gamma2 ~ dnorm(0, 0.01)       #Effect of region

#Alpha0
mu_a0 ~ dnorm(0, 0.01)        #Mean
tau_a0 ~ dgamma(0.1, 0.1)     #Precision
sig_a0 <- 1/sqrt(tau_a0)      #Variance

#Alpha1
mu_a1 ~ dnorm(0, 0.01)        #Mean
tau_a1 ~ dgamma(0.1, 0.1)     #Precision
sig_a1 <- 1/sqrt(tau_a1)      #Variance

#Beta1
mu_b1 ~ dnorm(0, 0.01)        #Mean
tau_b1 ~ dgamma(0.1, 0.1)     #Precision
sig_b1 <- 1/sqrt(tau_b1)      #Variance

#Overdispersion
r.N ~ dunif(0,100)            #Number of groups
r.G ~ dunif(0,100)            #Group size

for(s in social){

#Expected Group Size
beta0[s] ~ dunif(0,50)           #Intercept parameter
beta1[s] ~ dnorm(mu_b1, tau_b1)  #Effect parameter

} #end s loop

for(s in 1:nspec){

#Psi
tau_p[s] ~ dgamma(0.1, 0.1)  #Precision
sig_p[s] <- 1/sqrt(tau_p[s]) #Variance

#Sigma
gamma0[s] ~ dnorm(mu_s, tau_s)  #Intercept parameter

#Expected Number of Groups
```

```
alpha0[s] ~ dnorm(mu_a0, tau_a0)    #Intercept parameter
alpha1[s] ~ dnorm(mu_a1, tau_a1)    #Effect parameter

for(j in 1:nsites){

psi[j,s] ~ dnorm(0, tau_p[s])       #Transect effect parameter

#Scale parameter
sigma[j,s] <- exp(gamma0[s] + gamma1 * size[s] + gamma2 * region[j])

#------------#
#-LIKELIHOOD-#
#------------#

for(t in 1:nreps[j]){

#Construct cell probabilities for nG cells using numerical integration
#Sum of the area (rectangles) under the detection function

for(k in 1:nD){

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

#Proportion of each interval (width of rectangle) for both sides of the transect
pi[k,t,j,s] <- v/B

#Detection probability for each distance class k (area of each rectangle)
f[k,t,j,s] <- g[k,t,j,s] * pi[k,t,j,s]

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

}#end k loop

#Detection probability at each transect (sum of rectangles)
pcap[t,j,s] <- sum(f[1:nD,t,j,s])

#Observed population @ each t,j,s (N-mixture)
y[t,j,s] ~ dbin(pcap[t,j,s], N[t,j,s])

#Latent Number of Groups @ each t,j,s (negative binomial)
N[t,j,s] ~ dpois(lambda.star[t,j,s])

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

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

#Linear predictor for Expected Number of Groups
lambda[t,j,s] <- exp(alpha0[s] + alpha1[s] * region[j] + psi[j,s] + log(offset[j]))

}#end t loop
```

```
#Mean detection probability @ each j,s
psite[j,s] <- mean(pcap[1:nreps[j], j, s])

}#end j loop

#Mean detection probability for each species
Dprop[s] <- mean(psite[1:nsites, s])

}#end s loop

#Mean detection probability for all t,j,s
TotalDprop <- mean(Dprop[])

for(s in social){

for(j in 1:nsites){

for(t in 1:nreps[j]){

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

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

#Linear predictor for Expected Group Size
gs.lam[t,j,s] <- exp(beta0[s] + beta1[s] * region[j] + log(offset[j]))

}#end t loop

}#end j loop

}#end s loop

for(i in 1:nobs){

#Observed distance classes
dclass[i] ~ dcat(fc[1:nD, rep[i], site[i], spec[i]])

}#end i loop

for(i in 1:nsoc){

#Observed Group Size (zero truncated negative binomial)
gs[i] ~ dpois(gs.lam.star[s.rep[i], s.site[i], s.spec[i]]) T(1,)

}#end i loop

for(s in social){

for(j in 1:nsites){

for(t in 1:nreps[j]){
```

```
#Abundance per transect
GSrep[t,j,s] <- lambda.star[t,j,s] * gs.lam.star[t,j,s]

}#end t loop

#Abundance per transect averaged over surveys
GSsite[j,s] <- mean(GSrep[1:nreps[j], j, s])

}#end j loop

#Mean abundance per transect
GS[s] <- mean(GSsite[1:nsites, s])

#Abundance per transect for each region
RegGS[s,1] <- mean(GSsite[1:13, s])     #Mara Triangle
RegGS[s,2] <- mean(GSsite[14:17, s])    #Talek region

}#end s loop

}#end model
```

Appendix C: Model Results

Summary (mean, standard deviation, 95% credible interval [CI]) of parameter estimates from hierarchical multi-species distance sampling model. μ_σ: mean of $\gamma 0_s$; τ^2_σ: variance of $\gamma 0_s$; $\mu_{\alpha 0}$: mean of $\alpha 0_s$; $\tau^2_{\alpha 0}$: variance of $\alpha 0_s$; $\mu_{\alpha 1}$: mean of $\alpha 1_s$; $\tau^2_{\alpha 1}$: variance of $\alpha 1_s$; $\mu_{\beta 1}$: mean of $\beta 1_s$; $\tau^2_{\beta 1}$: variance of $\beta 1_s$; $\gamma 0_s$: species-specific intercepts of σ_{js}; $\alpha 0_s$: species-specific intercepts of λ_{tjs}; $\alpha 1_s$: species-specific effects of management regime on λ_{tjs}; $\beta 0_s$: species-specific intercepts of μ_{tjs}; $\beta 1_s$: species-specific effects of management regime on μ_{tjs}; $\gamma 1$: effect of body size on σ_{js}; $\gamma 2$: effect of management regime on σ_{js}; $Density_{s,MT}$: species-specific density in Mara Triangle; $Density_{s,TR}$: species-specific density in the Talek region; AL: African lion; BM: banded mongoose; BEF: bat-eared fox; BBJ: black-backed jackal; CAR: caracal; CHE: cheetah; LEO: leopard; SER: serval; SSJ: side-striped jackal; SM: slender mongoose; SH: spotted hyena.

Parameter	Mean	SD	2.5% CI	97.5% CI
μ_σ	4.13	0.22	3.66	4.53
τ^2_σ	0.6	0.25	0.27	1.2
$\mu_{\alpha 0}$	-1.33	0.59	-2.55	-0.2
$\tau^2_{\alpha 0}$	1.71	0.52	0.97	2.97
$\mu_{\alpha 1}$	-0.24	0.45	-1.21	0.58
$\tau^2_{\alpha 1}$	0.95	0.42	0.35	1.98
$\mu_{\beta 1}$	-0.65	0.37	-1.48	-0.03
$\tau^2_{\beta 1}$	0.65	0.36	0.24	1.59
$\gamma 0_{AL}$	3.98	0.59	2.82	5.18
$\gamma 0_{BM}$	4.28	0.17	3.95	4.61
$\gamma 0_{BEF}$	3.88	0.17	3.55	4.21
$\gamma 0_{BBJ}$	4.62	0.14	4.34	4.9
$\gamma 0_{CAR}$	3.77	0.56	2.61	4.79
$\gamma 0_{CHE}$	4.67	0.27	4.22	5.26
$\gamma 0_{LEO}$	3.6	0.53	2.49	4.57
$\gamma 0_{SER}$	4.03	0.25	3.56	4.56
$\gamma 0_{SSJ}$	4.4	0.32	3.84	5.09
$\gamma 0_{SM}$	3.56	0.38	2.8	4.25
$\gamma 0_{SH}$	4.59	0.1	4.41	4.79
$\alpha 0_{AL}$	-0.59	0.26	-1.14	-0.13
$\alpha 0_{BM}$	0.54	0.35	-0.21	1.13
$\alpha 0_{BEF}$	-1.87	1.08	-4.2	0.04
$\alpha 0_{BBJ}$	0.03	0.35	-0.74	0.63
$\alpha 0_{CAR}$	-3.05	1.12	-5.57	-1.04
$\alpha 0_{CHE}$	-2.73	0.62	-4.09	-1.67
$\alpha 0_{LEO}$	-2.71	0.99	-4.81	-0.89
$\alpha 0_{SER}$	-1.36	0.54	-2.56	-0.42
$\alpha 0_{SSJ}$	-2.91	0.79	-4.64	-1.57
$\alpha 0_{SM}$	-1.06	0.71	-2.5	0.33
$\alpha 0_{SH}$	0.99	0.21	0.55	1.38
$\alpha 1_{AL}$	-1.2	0.58	-2.38	-0.1
$\alpha 1_{BM}$	-0.19	0.53	-1.24	0.89
$\alpha 1_{BEF}$	-0.48	0.97	-2.57	1.33
$\alpha 1_{BBJ}$	0.63	0.53	-0.4	1.71
$\alpha 1_{CAR}$	-0.61	1.04	-3.05	1.12
$\alpha 1_{CHE}$	0.02	0.7	-1.39	1.39
$\alpha 1_{LEO}$	-0.73	1.02	-3.18	0.93
$\alpha 1_{SER}$	-0.68	0.79	-2.45	0.68
$\alpha 1_{SSJ}$	0.18	0.8	-1.38	1.81
$\alpha 1_{SM}$	-0.27	0.72	-1.81	1.07

Parameter	Mean	SD	2.5% CI	97.5% CI
$\alpha1_{SH}$	0.65	0.39	-0.13	1.43
$\beta0_{AL}$	1.23	0.14	0.95	1.51
$\beta0_{BM}$	2.43	0.12	2.2	2.66
$\beta0_{BEF}$	0.92	0.17	0.58	1.27
$\beta0_{BBJ}$	0.26	0.15	0.02	0.59
$\beta0_{CHE}$	0.33	0.28	0.01	1.02
$\beta0_{SM}$	0.37	0.29	0.01	1.07
$\beta0_{SH}$	0.11	0.08	0	0.31
$\beta1_{AL}$	-0.93	0.52	-2.07	-0.03
$\beta1_{BM}$	-0.4	0.22	-0.82	0.04
$\beta1_{BEF}$	-0.99	0.63	-2.45	0.03
$\beta1_{BBJ}$	-0.7	0.28	-1.27	-0.18
$\beta1_{CHE}$	-0.6	0.55	-1.78	0.4
$\beta1_{SM}$	-0.97	0.77	-2.84	0.19
$\beta1_{SH}$	0.03	0.17	-0.31	0.36
$\gamma1$	0.52	0.07	0.39	0.65
$\gamma2$	0.43	0.22	-0.01	0.87
$Density_{AL,MT}$	2.23	0.5	1.41	3.36
$Density_{BM,MT}$	28.21	5.21	19.17	39.65
$Density_{BEF,MT}$	7.73	1.77	4.75	11.63
$Density_{BBJ,MT}$	1.86	0.42	1.18	2.81
$Density_{SH,MT}$	3.49	0.45	2.71	4.46
$Density_{AL,TR}$	0.35	0.24	0.07	0.95
$Density_{BM,TR}$	24.37	6.4	14.06	39.1
$Density_{BEF,TR}$	0.66	0.57	0.07	2.15
$Density_{BBJ,TR}$	2.4	0.69	1.29	3.95
$Density_{SH,TR}$	10.63	1.53	7.92	13.86