Supplementary Information

# Supplementary Information 1: Code to generate lion density estimates reported in Rangers on the Frontline of Wildlife Monitoring: African Lions in Uganda’s Nile Delta

*23 August 2024*

Here is the code for you to run the analysis. Start off by making 1 folder called Murchison Falls Nile Delta lion survey. Then within this folder make 4 separate folders called model 1, model 2, model 3, model 4. Then copy the input files (all files labelled sup data ie.

Capture history (supp data 2), search effort (supp data 4), habitat file (supp data 1), traps (supp data 3) and sex denoted as 0 and 1 for males and females respectively (supp data 5) and dump these and all function files (ie. SCRi.fn.par1- lionVer1003.R, scrdataWS.R, and e2dist.R) into each of these (ie. model 1, model 2, model 3, model 4.).

Once you have done all of the above steps, copy and paste all of the code below into 4 separate R sessions (depending on how many CPU cores your remote desktop or server features….if CPU usage is minor, then you can run all models, if not you may have to run them individually). Importantly, your directory names will be a little different to what is listed below (ie. your path names). Everything else should be the same.

Essentially the only thing we are tweaking each time with this code is the file path (ie. telling the model where to store our results) and the model configuration (in our secrbayes analysis we will only run 4 models).

Now the most important thing for you to do is open 4 separate R studio sessions and run each respective model on each of these sessions. The good thing is that with this code you will have 4 chains printed out by R in your individual model folders (ie. model 1, model 2 etc.).

Once you have all the chains in your folders (which I anticipate will be anything from 1-5 days of running on your computer) we can finalise all of the results and model selections.

Specifically, we want:

11000 iterations

1000 burn in

You can also do a few test runs on each model…just to see if the model is running and storing results in the correct folder! To do this change niter from 10000 to 100, and burn from 1000 to 10 – the null model will look like this:

Lionnull <- SCRi.fn.par1(scrmurchisondata, contin = contin, modelno=1, nc=nchains, ni=niter, burn=1000, skip=1, nz=1000, theta=1, Msigma=1, Mb=0, Msex=0, Msexsigma=0, Xsex=Xsex, Xeff=Xeffort, Xeff1=NULL, Xeff2=NULL, Xeff3=NULL, ss.prob=NULL, coord.scale=1000, area.per.pixel=1, thinstatespace=1, maxNN=40, dumprate=1000)

If everything goes well R should accept the code and you should see 4 new folders in your Model 1 folder. If not…troubleshoot and see if you can fix the error. If not get back to me.

**Murchison Falls Nile Delta Lion Survey Data**

**Model 1 Murchison Falls Lions**

dirMain = setwd("/Users/aleksbraczkowski/desktop/Murchison R/Murchison Input Files/Model 1")

library(doSNOW)

statespace <- read.csv("supp data 1.csv")

traps <- read.csv("supp data 3.csv")

captures <- read.csv("supp data 2.csv")

sex <- read.csv("supp data 5.csv")

effort <- read.csv("supp data 4.csv")

Xsex <- sex[,2]

Xeffort <- effort[,4:ncol(effort)]

Xeffort <- sapply(Xeffort, as.numeric)

Xeffort1 <- NULL

Xeffort2 <- NULL

Xeffort3 <- NULL

alive <- matrix(1, nrow = length(unique(captures[,"ANIMAL\_ID"])), ncol = ncol(Xeffort))

source("e2dist.R")

source("SCRi.fn.par1-lionVer1003.R")

source("scrDataWS.R")

scrmurchisondata <- scrData(traps=traps, captures=captures, statespace=statespace, Xsex=Xsex, Xeff=Xeffort, Xeff1=Xeffort1, Xeff2=Xeffort2, Xeff3=Xeffort3)

niter <- 11000

nchains <- 4

modelno <- 1

contin <- 0

Lionnull <- SCRi.fn.par1(scrmurchisondata, contin = contin, modelno=1, nc=nchains, ni=niter, burn=1000, skip=1, nz=1000, theta=1, Msigma=1, Mb=0, Msex=0, Msexsigma=0, Xsex=Xsex, Xeff=Xeffort, Xeff1=NULL, Xeff2=NULL, Xeff3=NULL, ss.prob=NULL, coord.scale=1000, area.per.pixel=1, thinstatespace=1, maxNN=40, dumprate=1000)

**Model 2 Murchison Falls Lions**

dirMain = setwd("/Users/aleksbraczkowski/desktop/Murchison R/Murchison Input Files/Model 2")

library(doSNOW)

statespace <- read.csv("supp data 1.csv")

traps <- read.csv("supp data 3.csv")

captures <- read.csv("supp data 2.csv")

sex <- read.csv("supp data 5.csv")

effort <- read.csv("supp data 4.csv")

Xsex <- sex[,2]

Xeffort <- effort[,4:ncol(effort)]

Xeffort <- sapply(Xeffort, as.numeric)

Xeffort1 <- NULL

Xeffort2 <- NULL

Xeffort3 <- NULL

alive <- matrix(1, nrow = length(unique(captures[,"ANIMAL\_ID"])), ncol = ncol(Xeffort))

source("e2dist.R")

source("SCRi.fn.par1-lionVer1003.R")

source("scrDataWS.R")

scrmurchisondata <- scrData(traps=traps, captures=captures, statespace=statespace, Xsex=Xsex, Xeff=Xeffort, Xeff1=Xeffort1, Xeff2=Xeffort2, Xeff3=Xeffort3)

niter <- 11000

nchains <- 4

modelno <- 2

contin <- 0

LeopsigmaSex <- SCRi.fn.par1(scrmurchisondata, contin = contin, modelno=2, nc=nchains, ni=niter, burn=1000, skip=1, nz=1000, theta=1, Msigma=1, Mb=0, Msex=0, Msexsigma=1, Xsex=Xsex, Xeff= Xeffort, Xeff1=NULL, Xeff2=NULL, Xeff3=NULL, ss.prob=NULL, coord.scale=1000, area.per.pixel=1, thinstatespace=1, maxNN=40, dumprate=1000)

**Model 3 Murchison Falls Lions**

dirMain = setwd("/Users/aleksbraczkowski/desktop/Murchison R/Murchison Input Files/Model 3")

library(doSNOW)

statespace <- read.csv("supp data 1.csv")

traps <- read.csv("supp data 3.csv")

captures <- read.csv("supp data 2.csv")

sex <- read.csv("supp data 5.csv")

effort <- read.csv("supp data 4.csv")

Xsex <- sex[,2]

Xeffort <- effort[,4:ncol(effort)]

Xeffort <- sapply(Xeffort, as.numeric)

Xeffort1 <- NULL

Xeffort2 <- NULL

Xeffort3 <- NULL

alive <- matrix(1, nrow = length(unique(captures[,"ANIMAL\_ID"])), ncol = ncol(Xeffort))

source("e2dist.R")

source("SCRi.fn.par1-lionVer1003.R")

source("scrDataWS.R")

scrmurchisondata <- scrData(traps=traps, captures=captures, statespace=statespace, Xsex=Xsex, Xeff=Xeffort, Xeff1=Xeffort1, Xeff2=Xeffort2, Xeff3=Xeffort3)

niter <- 11000

nchains <- 4

modelno <- 3

contin <- 0

LeoplambdaSex <- SCRi.fn.par1(scrmurchisondata, contin = contin, modelno=3, nc=nchains, ni=niter, burn=1000, skip=1, nz=1000, theta=1, Msigma=1, Mb=0, Msex=1, Msexsigma=0, Xsex=Xsex, Xeff= Xeffort, Xeff1=NULL, Xeff2=NULL, Xeff3=NULL, ss.prob=NULL, coord.scale=1000, area.per.pixel=1, thinstatespace=1, maxNN=40, dumprate=1000)

**Model 4 Murchison Falls Lions**

dirMain = setwd("/Users/aleksbraczkowski/desktop/Murchison R/Murchison Input Files/Model 4")

library(doSNOW)

statespace <- read.csv("supp data 1.csv")

traps <- read.csv("supp data 3.csv")

captures <- read.csv("supp data 2.csv")

sex <- read.csv("supp data 5.csv")

effort <- read.csv("supp data 4.csv")

Xsex <- sex[,2]

Xeffort <- effort[,4:ncol(effort)]

Xeffort <- sapply(Xeffort, as.numeric)

Xeffort1 <- NULL

Xeffort2 <- NULL

Xeffort3 <- NULL

alive <- matrix(1, nrow = length(unique(captures[,"ANIMAL\_ID"])), ncol = ncol(Xeffort))

source("e2dist.R")

source("SCRi.fn.par1-lionVer1003.R")

source("scrDataWS.R")

scrmurchisondata <- scrData(traps=traps, captures=captures, statespace=statespace, Xsex=Xsex, Xeff=Xeffort, Xeff1=Xeffort1, Xeff2=Xeffort2, Xeff3=Xeffort3)

niter <- 11000

nchains <- 4

modelno <- 4

contin <- 0

LeoplambdasigmaSex <- SCRi.fn.par1(scrmurchisondata, contin = contin, modelno=4, nc=nchains, ni=niter, burn=1000, skip=1, nz=1000, theta=1, Msigma=1, Mb=0, Msex=1, Msexsigma=1, Xsex=Xsex, Xeff= Xeffort, Xeff1=NULL, Xeff2=NULL, Xeff3=NULL, ss.prob=NULL, coord.scale=1000, area.per.pixel=1, thinstatespace=1, maxNN=40, dumprate=1000)

**Parameter Estimation Code**

Code to generate parameter estimates and assess model diagnostics (after models have run and chain outputs have been pasted in your model directories)

When your analysis is complete every folder (eg. Model 1) will contain the written model outputs in the form of individual chains (there will be 4 chains, so 4 folders). Now you have to derive the diagnostics to 1) examine if the models have converged, and 2) to get the parameter estimates. The code below will help you to achieve this. Note the code is written in a way so that you can paste the results from the R terminal (ie. copy and paste the outputs generated by the code). You can also copy over the image of the diagnostic plots illustrating model correlations. Note, you will do this for each of the four models – the code below is an illustration for model 1 only:

**Model 1 Murchison Lions**

First take all of the outputs in each of the chains written in your respective model folders and dump them into one folder named “Outputs” (ie. just copy everything in each chain mcmc output folder and paste it into the “Outputs” folder – set your new working directory to this folder once you have done this) – now run this code:

### Code to do MCMC diagnostics and calculate posterior HPDs of estimates in Bayesian SECR

### For Murchison

## Call coda package (for MCMC diagnostics) and mcmcse package (to compute Monte Carlo error)

library(coda) library(mcmcse) library(parallel)

### Obtain all the MCMC histories. These directory structures have to be replaced by the ones

obtained after running the analysis ###

histCH1 <- read.csv("SHOmcmchist\_230507\_155258CH2.csv") gdataCH1 <- read.csv("gofdata\_230507\_155258CH2.csv") gnewCH1 <- read.csv("gofnew\_230507\_155258CH2.csv")

### Activate these only on good computers ####

AcCentresCH1 <- read.csv("AcCentres\_230507\_155258CH2.csv") RealIndividualsCH1 <-read.csv("RealIndividuals\_230507\_155258CH2.csv")

histCH2 <- read.csv("SHOmcmchist\_230507\_155036CH3.csv") gdataCH2 <- read.csv("gofdata\_230507\_155036CH3.csv")

gnewCH2 <- read.csv("gofnew\_230507\_155036CH3.csv") AcCentresCH2 <- read.csv("AcCentres\_230507\_155036CH3.csv")

RealIndividualsCH2 <- read.csv("RealIndividuals\_230507\_155036CH3.csv")

histCH3 <- read.csv("SHOmcmchist\_230507\_154753CH1.csv") gdataCH3 <- read.csv("gofdata\_230507\_154753CH1.csv") gnewCH3 <- read.csv("gofnew\_230507\_154753CH1.csv") AcCentresCH3 <- read.csv("AcCentres\_230507\_154753CH1.csv")

RealIndividualsCH3 <- read.csv("RealIndividuals\_230507\_154753CH1.csv")

histCH4 <- read.csv("SHOmcmchist\_230507\_154142CH4.csv") gdataCH4 <- read.csv("gofdata\_230507\_154142CH4.csv") gnewCH4 <- read.csv("gofnew\_230507\_154142CH4.csv") AcCentresCH4 <- read.csv("AcCentres\_230507\_154142CH4.csv")

RealIndividualsCH4 <- read.csv("RealIndividuals\_230507\_154142CH4.csv") #### Create MCMC objects ####

histCH1mcmc <- as.mcmc(histCH1) histCH2mcmc <- as.mcmc(histCH2) histCH3mcmc <- as.mcmc(histCH3) histCH4mcmc <- as.mcmc(histCH4)

## Remove beta.behave column, X column(iter no) and Density(for a strange reason gives an error)

and set start and end for extended burnin start<-1

end<-10000

histCH1mcmc <- window(histCH1mcmc[,c(-1,-7)], start,end) histCH2mcmc <- window(histCH2mcmc[,c(-1,-7)], start,end) histCH3mcmc <- window(histCH3mcmc[,c(-1,-7)], start,end) histCH4mcmc <- window(histCH4mcmc[,c(-1,-7)], start,end)

### Combine chain outputs ###

combinedHist <- rbind(histCH1mcmc, histCH2mcmc, histCH3mcmc, histCH4mcmc) chainList <- list(histCH1mcmc, histCH2mcmc, histCH3mcmc, histCH4mcmc)

## MCMC diagnostics ##

## Multi-chain convergence check using Gelman-Rubin diagnostic

gelmandiag <- gelman.diag(chainList, confidence=FALSE, transform=FALSE, autoburnin=FALSE,

multivariate=FALSE)

## Single chain convergence check using Geweke diagnostic (optional) gewekediag <- geweke.diag(histCH1mcmc)

#### Report MCMC diagnostic results. For Geweke we want the magnitude (-ve or +ve) for each parameter to be less than 1.64. For Gelman-Rubin we want Potential Shrink Reduction Factor to be

less than 1.2 (1.1 or lower for more defensible runs) for each parameter. gelmandiag

gewekediag

### Summary results. Look for how different median is to the mean. This indicates nature of the

posterior distribution. Ideally we would like them to be nearly the same. But OK otherwise too.

mean.model1 <- apply(combinedHist,2,mean)

## Obtain mean of the estimates with the Monte Carlo error mean.model1 <- mcse.mat(combinedHist, method="bm", g=NULL) sd.model1 <- apply(combinedHist,2,sd)

mean.model1 sd.model1

## Highest posterior density intervals for one of the chains # This piece of code is taken from SPACECAP version 1.1.0 (Gopalaswamy et al. 2015) ##

HPDinterval(histCH1mcmc)

#### Goodness-of-fit statistics #### ### Obtain all the gof statistics ###

### Combine gdata and gnew ###

gdatacombined <- rbind(gdataCH1, gdataCH2, gdataCH3, gdataCH4) gnewcombined <- rbind(gnewCH1, gnewCH2, gnewCH3, gnewCH4)

## Bayesian p-value calculation ##

BayesPval <- mean(gdatacombined[,2]>gnewcombined[,2])

BayesPval

## Generate pair-wise plots. This will be useful for assessing estimation covariances and parameter

redundancies (if any) owing to poor sample sizes ## pairs(combinedHist, gap=0, pch=".")

### Generate pixel-specific density estimates ###

## Combine activity centres and real individuals file into a combined history ##

AcCentresCombined <- rbind(AcCentresCH1, AcCentresCH2, AcCentresCH3, AcCentresCH4)

RealIndividualsCombined <- rbind(RealIndividualsCH1, RealIndividualsCH2, RealIndividualsCH3,

RealIndividualsCH4)

## Obtain the unscaled statespace (any one chain is sufficient as it comes from input data) ## SSunscaledCH <- read.csv("SSunscaled\_230506\_122647CH2.csv")

nG <- nrow(SSunscaledCH)

# Set pixel ID of home range centers for phantom animals to zero indlocsCH1 <- AcCentresCH1 \* RealIndividualsCH1 indlocsCH2 <- AcCentresCH2 \* RealIndividualsCH2 indlocsCH3 <- AcCentresCH3 \* RealIndividualsCH3 indlocsCH4 <- AcCentresCH4 \* RealIndividualsCH4

indlocs <- rbind(indlocsCH1, indlocsCH2, indlocsCH3, indlocsCH4) indlocnum <- data.matrix(indlocs)

# Count the proportion of times each pixel was a home range centre,

# convert to animals per sq km (here 1 sq km was input data for elephant analysis - so change

accordingly)

densVec <- tabulate(indlocnum, nbins=nG) / nrow(indlocs) / 0.336

dirMain <- "/Users/s2990525/Desktop/R ANALYSIS/Murchison Lions small buffer/Model 1/All"

setwd(dirMain)

GEC\_Loango\_CAMTRAP\_SS <- read.csv("Habitat.csv") pixelDensity <- GEC\_Loango\_CAMTRAP\_SS pixelDensity$`Pixel Density` <- GEC\_Loango\_CAMTRAP\_SS[, 3]

pixelDensity$`Pixel Density`[GEC\_Loango\_CAMTRAP\_SS[, 3] > 0] <- densVec

# Generate csv file for pixel densities #

nameoffile3 = paste(dirMain,"/GEC\_Loango\_CAMTRAP\_PixelDens.csv", sep="") write.csv(pixelDensity, file=nameoffile3)

# This part is to obtain posterior standard deviations on pixel-specific densities #

# Create an abundance matrix of dimension no. of iterations x total number of grid cells # abundMatrix <- matrix(data=NA, nrow=nrow(indlocs), ncol=nG)

# Fill up this matrix with abundance counts for each iteration # for (i in 1:nrow(indlocs)){

abundVecTemp <- tabulate(indlocnum[i,], nbins=nG) abundMatrix[i,] <- abundVecTemp

}

# This part is meant to compute abundances for sub-regions #

# Enter the sequence of grid cell numbers for analysis. This will be a selection of numbers between 1

to nG corresponding to which cells are being analysed. For the entire study area this can simply be

1:nG. In the example below, it indicates that only grid cells 1,3,5,7 are chosen for reporting.

This will be according to the sub-region chosen #

gridVec <- c(1:nG)

# Obtain total abundance counts for grid cells referenced by gridVec for each iteration # abundVecTotal <- rowSums(abundMatrix[,gridVec])

# Obtain posterior mean and standard deviations of the sub-region meanAbund <- mean(abundVecTotal)

sdAbund <- sd(abundVecTotal) meanAbund

sdAbund

Marginal Likelihood Estimation

##### R code snippet to estimate the marginal likelihood and its associated standard deviation from

LogLikelihood outputs in SECR #####

rm(list = ls()) options(digits = 8)

source('BMSE.utility.functions.1.R') start.time = Sys.time()

ts = format(Sys.time(), "%d%m%y\_%H%M%S")

#=================================================

# Harmonic mean estimator of marginal likelihood #=================================================

# g(mu, L) = pi(mu, L) i.e Takng prior of (mu, L) as the tuning density of (mu, L) #loglik.chain = unlist(read.csv(paste0(folderpath, '/markovchain.loglikelihood.txt', sep = ''), sep = ',',

header = T))[(burnin + 1):ndraws] #if(model == 1 | model == 2)

# {

# logfactor.sex = log(post.theta^(post.z[,1:numl]\*post.sex[,1:numl])) + log((1 - post.theta)^(1 -

post.z[,1:numl]\*post.sex[,1:numl])) # tot.length x numl

# loglik.chain = loglik.chain + rowSums(logfactor.sex) # tot.length x 1 # }

###### Read the LogLikelihood vector from the output file these will be in your outputs folder copy

them below and make sure the names are correct ######

logCH1 <- read.csv("LogLikelihood\_230507\_154142CH4.csv") logCH2 <- read.csv("LogLikelihood\_230507\_154753CH1.csv") logCH3 <- read.csv("LogLikelihood\_230507\_155036CH3.csv") logCH4 <- read.csv("LogLikelihood\_230507\_155258CH2.csv")

combinedlike <- rbind(logCH1, logCH2, logCH3, logCH4)

loglik.file <- combinedlike loglik.chain <- loglik.file[,2]

logh.chain = - loglik.chain # -loglik.zx0s.chain C = mean(logh.chain)

logmarglik.hm = gdmean3(logh.chain) tot.length = length(loglik.chain) sd.mhm = gdsd(logh.chain)

cat('Log of the estimated Marginal Likelihood using HM method =', logmarglik.hm) Log of the estimated Marginal Likelihood using HM method = -46222.86

You can now copy over the results from your terminal into the below template. Then make decisions on every model based on the details therein. The below is simply an example template, as are the results!

Gelmandiag

Potential scale reduction factors: Point est. Upper C.I.

|  |  |  |
| --- | --- | --- |
| bsigma | 1.04 | 1.03 |
| sigma | 1.07 | 1.06 |
| bsigma2 | 1.04 | 1.03 |
| sigma2 | 1.07 | 1.06 |
| lam0 | 1.03 | 1.03 |
| beta1.effort. | 1.07 | 1.06 |
| beta2.effort. | NaN | NaN |
| beta3.effort. | 1.01 | 1.01 |
| beta4.effort. | NaN | NaN |
| beta.sex | NaN | NaN |
| psi | 1.02 | 1.02 |
| psi.sex | 1.00 | 1.00 |
| Nsuper | 1.03 | 1.02 |
| theta | NaN | NaN |
| beta.density | NaN | NaN |
| D | 1.03 | 1.02 |
| D.adj | 1.03 | 1.02 |
| gewekediag |  |  |

Fraction in 1st window = 0.1 Fraction in 2nd window = 0.5

bsigma sigma bsigma2 sigma2 lam0 1.5242399 -2.0166206 1.5242399 -2.0166206 2.0783149

beta1.effort. beta2.effort. beta3.effort. beta4.effort. beta.sex

-3.3733703 NaN -0.4904505 NaN NaN

psi psi.sex Nsuper theta beta.density 1.7875086 0.0002677 2.0444104 NaN NaN

D D.adj

2.0444104 2.0444104

mean.model1 est se

bsigma 0.035281066 2.319348e-03

sigma 4.254270189 1.999733e-01

bsigma2 0.035281066 2.319348e-03

sigma2 4.254270189 1.999733e-01

lam0 0.001670143 7.939997e-05

beta1.effort. 4.646722425 1.253828e-01

beta2.effort. 0.000000000 0.000000e+00

beta3.effort. 4.736051145 3.887527e-02

beta4.effort. 0.000000000 0.000000e+00

beta.sex 0.000000000 0.000000e+00

psi 0.397474608 1.095034e-02

psi.sex 0.596193370 2.581765e-03

Nsuper 41.496400000 1.186475e+00

theta 1.000000000 0.000000e+00

beta.density 0.000000000 0.000000e+00

D 0.008814588 2.520288e-04

D.adj 0.009123906 2.608729e-04

sd.model1

bsigma sigma bsigma2 sigma2 lam0

1.864966e-02 1.392919e+00 1.864966e-02 1.392919e+00 9.841158e-04

beta1.effort. beta2.effort. beta3.effort. beta4.effort. beta.sex 1.045552e+00 0.000000e+00 6.348596e-01 0.000000e+00 0.000000e+00

psi psi.sex Nsuper theta beta.density

1.951652e-01 2.127742e-01 2.035795e+01 0.000000e+00 0.000000e+00

D D.adj 4.324399e-03 4.476149e-03

HPDinterval(histCH1mcmc) lower upper

bsigma 1.129272e-02 0.077416583

sigma 2.262342e+00 5.473618499

bsigma2 1.129272e-02 0.077416583

sigma2 2.262342e+00 5.473618499

lam0 4.169666e-04 0.003817648

beta1.effort. 2.697444e+00 6.172672343

beta2.effort. 0.000000e+00 0.000000000

beta3.effort. 3.482332e+00 5.706961997

beta4.effort. 0.000000e+00 0.000000000

beta.sex 0.000000e+00 0.000000000

psi 1.262590e-01 0.850673786

psi.sex 2.012784e-01 0.954461412

Nsuper 1.300000e+01 88.000000000

theta 1.000000e+00 1.000000000

beta.density 0.000000e+00 0.000000000

D 3.398690e-03 0.019330050

D.adj 3.298083e-03 0.019788501

attr(,"Probability") [1] 0.95

BayesPval [1] 0.53315

# Supplementary Information 2: Specifications of SOLARIS WEAPON 4K infrared camera trap used in the collection of African lion photographic detections (Rangers on the frontline of wildlife monitoring: African lions in Uganda’s Nile Delta)

Lens specifications: 4.0mm focal length, F/NO=1.6, and an 89° field of view. 8MP resolution, with a maximum of 46MP through interpolation.

4K ultra-high-definition video

Trigger speed = 0.3 seconds (0.2 S with side PIR assistance)

Multi shot of up to 5 images per detection event Adjustable video length

Solar panel charging means you don't have to change batteries - even in dense tropical rainforests (southerly direction isn't a problem)

Space for 4x external alkaline or lithium AA batteries in case main battery pack fails

Lithium battery can operate in temperatures as low as -20°C and in excess of 40°C (so will work fine in your site)

WiFi functionality allows for remote photo and video viewing and downloading.

36 infrared flashlights, giving you up to 20 meters (850nm) of coverage.

PIR (Pyroelectric Infrared) detection angle covers 120 degrees.

The central PIR detection angle is 60°, while each side PIR has a 30° detection angle

# Supplementary Information 3: Calculations Associated with Effort Costs for Rangers on the Frontline

1. To calculate the cost per unit effort for remote camera traps we implemented the following steps:
2. Applied a 1 sigma buffer around our outermost camera traps – we used this total area (encompassing traps) to provide a unit of effectively sampled area by traps. This area was equal to 255.60 km².
3. We then calculated the cost of the survey per unit of km² by dividing the number of trap nights by this effectively sampled area (i.e., 1601 trap nights/255.60 km²).
4. This results in a figure of 6.26 trap nights/km².
5. We then multiplied this 6.26 trap nights/km² by the costs associated with running the camera trap survey – these are the cumulative total of 1) salary of 1 research assistant to run the camera traps during the period (USD$), 2) the initial cost of 64 camera traps divided by 3 as traps are likely to last multiple surveys (USD$), the costs of diesel fuel to check camera traps during the survey period – in our case traps were checked seven times (USD$ 227.60). Each of these camera checks required 144.97 km of driving. Cumulative total of survey cost = USD$3259.77/1601 trap nights.
6. This gave us a total of USD$2.04/trap night/km².
7. Finally, we divided the number of lion detections (total), and only usable by the number of trap nights.
8. These two calculations gave us a grand total of USD$ needed to record Lion Detections/Trap Night/km².

To calculate the cost per unit effort for our ranger-based search encounter lion survey we implemented the following steps:

1. For consistency we used the same 1 sigma buffer around our camera traps to define the effectively sampled area.
2. We then divided the total number of kilometers driven in the area by rangers by the effectively sampled area (2939 km/255.60 km²=11.50).
3. This figure was then divided by the number of sampling days recorded by rangers (i.e., 60 days upon which sampling took place) to derive search effort/day/km² = 11.5/60=0.19 km/day/km².
4. We then multiplied this 0.19 km expended/day/km² by the costs associated with running the search encounter survey – these are the cumulative total of 1) top up salaries of the two rangers, LN, and SM to collect data during the survey (USD$), 2) the equipment cost of two cameras and two secondhand iPhone 8 smartphones (USD$), and 3) the costs of training rangers to collect data. Cumulative total of survey cost = USD$1633.66/2939 km search effort expended.
5. This gave us a total of USD$0.55/search effort day/km².