Kodiak Mountain Goat RSF

Inventory and Monitoring Program
Region 7 (Alaska)

McCrea Cobb

June 2017

Abstract

This is the abstract.

Contents

Backgound

Principal Investigators

Bill Pyle

Association

Kodiak National Wildlife Refuge

Email Address

bill_pyle@fws.gov

Type of Request

Data analysis

Project Objective

(A sentence or two specifying the objective of the work)

Project Description

(A short paragraph describing the project)

PRIMR Number

(Survey ID from the PRIMR database)

Methods

Data import and formatting

GPS collar fixes

I imported raw ATS and Telonics collar data into R using the "ImportFormat.R" script. I reformated the dataset for RSF analysis. This included creating a data (POSIXct) variable, an dummy variable ("Response") indicating whether it was a used or available location, and a variable indicating the collar type (ATS/Telonics). I merged the Telonics and ATS datasets and created a spatial points data frame (UTM NAD83, Zone 5N) of the result.

Covariates

Defining habitat availablity

Population level

I measured resource selection (use vs. availablity design) at the population level (Johnson (1980) second order selection). I defined the extent of habitat availability using a 99% kernel home range of fixes from all collared goats. Within this extent I generated a random sample of available fixes at a 1:10 (used:available) (Northrup et al. 2013).

Individual level

Method B. Resource selection function (RSF), individual level

b. Individual level

    1. Buffer each point by the mean step length ("available" region), generate 100 "available" points within each of these region.

Model selection

We compared covariate values at used locations to those at available locations. Individual collared mountain goats were the sampling unit. For the population level analysis, we evaluated the relative probability of use with binomial (logit link) mixed effects models, with a random intercept for each collared mountain goat to account for the unbalanced design and correlation among individuals. For the individual level models, we used

I followed a two-tiered approach to model selection. In the first tier, I fit univariate models of terrian covariates (slope, elevation, and vector ruggedness measure) that were derived from USGS NEDs. I used AICc for model selection. In the second tier, I began with the top candidate terrian model as a base model and then evaluated all combinations of habitat type covariates, which included Forest, Shrubs, Tundra/Heath, Meadow, Water, Snow and Rock. I derived these habitat covariates from a habitat classification raster layer based on ground-truthed LandSat data (cite). All rasters had a spatial scale of 30-m and were snapped to the NED.

Individual level

Condition logistic regression (individual level analysis that can be extrapolated to the population level using parameter bootstrap of the individual model coefficients).

Results

Discussion

Appendix

R session info

```
## R version 3.3.0 (2016-05-03)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.15.1
##
## loaded via a namespace (and not attached):
##  [1] backports_1.0.5 magrittr_1.5    rprojroot_1.2   tools_3.3.0
##  [5] htmltools_0.3.6 yaml_2.1.14     Rcpp_0.12.10    stringi_1.1.5
##  [9] rmarkdown_1.5   stringr_1.2.0   digest_0.6.12   evaluate_0.10
```

R code for importing and formating ATS and Telonics GPS collar data

```
### 1. Import and and format the ATS data ###

# First, you need to save downloaded .txt data file as "R\Data\GPSData.txt".

dfATS <-read.table("./Data/ATSCollarData.txt", header = TRUE, sep = ",")
SerialNum <- read.table("./Data/SerialNum.txt", header = TRUE, sep = ",")

colnames(dfATS) <- c("SerialNum", "Year", "Day", "Hour", "Lat", "Long",
                     "DOP", "NumSats", "FixTime", "Fix2D3D")
dfATS$Year <- factor(paste(20, dfATS$Year, sep =""))  # Convert Year to a factor
dfATS$SerialNum <- factor(dfATS$SerialNum)  # Convert SerialNum to a factor
dfATS$Fix2D3D <- factor(dfATS$Fix2D3D)  # Convert 2D3D to a factor
SerialNum$SerialNum <- factor(SerialNum$SerialNum)
dfATS <- merge(dfATS, SerialNum, by = "SerialNum")

dfATS$Date <- as.factor(format(strptime(dfATS$Day, format = "%j"),
                               format = "%m/%d"))
dfATS$Date <- as.factor(paste(dfATS$Date, dfATS$Year, sep = "/"))
dfATS$Date <- strptime(paste(dfATS$Date, dfATS$Hour, sep = " "), "%m/%d/%Y %H")
dfATS$Date <- as.POSIXct(dfATS$Date)

dfATS <- subset(dfATS, Date >= "2015-08-01")  # subset dates after collaring
dfATS <- unique(dfATS)  # removes any duplicate values

# Clean up:
dfATS$SerialNum <- NULL
dfATS$FixTime <- NULL
dfATS$NumSats <- NULL
rm(SerialNum)

# Add a dummy variable for used/avail:
dfATS$Response <- 1

# Add a variable for the type of collar (Telonics/ATS)
dfATS$Collar <- "ATS"


### 2. Import and format the Telonics collar data ###

dfTel <-read.table("./Data/TelonicsCollarData.txt", header = TRUE, sep = ",")

dfTel$Lat <- dfTel$GPS_Latitu  # Rename Lat
dfTel$GPS_Latitu <- NULL

dfTel$Long <- dfTel$GPS_Longit  # Rename Long
dfTel$GPS_Longit <- NULL

levels(dfTel$GPS_Fix_At) <- c("2", "3")  # Format and rename "Fix2D3D"
dfTel$Fix2D3D <- dfTel$GPS_Fix_At
dfTel$GPS_Fix_At <- NULL
```

```r
dfTel$GPS_Horizo <- NULL  # Rename pdop
dfTel$Pdop <- dfTel$GPS_Positi

dfTel$GPS_UTM_Zo <- NULL  # Clean up
dfTel$GPS_UTM_No <- NULL
dfTel$GPS_UTM_Ea <- NULL
dfTel$GPS_Altitu <- NULL
dfTel$GPS_Speed <- NULL
dfTel$Predeploym <- NULL
dfTel$GPS_Headin <- NULL
dfTel$GPS_Positi <- NULL
dfTel$GPS_Satell <- NULL
dfTel$GPS_Sate_1 <- NULL
dfTel$GPS_Naviga <- NULL
dfTel$FID <- NULL

# Format Date:
dfTel$GPS_Fix_Da <- substr(dfTel$GPS_Fix_Da, 1, 9)  # remove the time slot from the dates
dfTel$Date <- strptime(paste(dfTel$GPS_Fix_Da, dfTel$GPS_Fix_Ti, sep = " "), "%m/%d/%Y %H:%M:%S")
dfTel$Date <- as.POSIXct(dfTel$Date)
# Round to the nearest hour:
dfTel$Date <- format(round(dfTel$Date, units = "hours"), format = "%Y-%m-%d %H:%M:%S")
dfTel$GPS_Fix_Da <- NULL
dfTel$GPS_Fix_Ti <- NULL

# Add a dummy variable for used/avail:
dfTel$Response <- 1

# Add a variable for the type of collar (Telonics/ATS)
dfTel$Collar <- "Telonics"

# Censor fixes that were collected after the collar was retrieved:
dfTel1 <- subset(dfTel, Date <= "xx-xx-xx" | CollarID == "xx")




### 3. Merge the ATS and Telonics data:
df <- merge(dfATS, dfTel, all = T)
rm(dfTel, dfATS)

# Subset df to dates after collaring:
df <- subset(df, Date >= "2015-08-01")

# Save it as a .csv:
write.csv(df, "./Data/df.csv", row.names = F)



## 4. Create a SpatialPointsDataFrame from the merged df:
dfSp <- SpatialPointsDataFrame(df[ ,2:1], df,
                               proj4string = CRS("+proj=longlat +datum=WGS84"))
# Reproject into UTM NAD83, Zone 5N:
```

```r
dfSp <- spTransform(dfSp, CRS("+proj=utm +zone=5 +datum=NAD83"))
plot(dfSp)  # Look at it
```

R code for formating spatial covariate data for RSF

```
### STEP 1. Import and format spatial covariate data ###

# Allow for parallel processing to speed things up:
require(snow)
require(parallel)
beginCluster( detectCores() - 1)

# Required packages:
library(rgdal)
library(maptools)
library(raster)
library(rasterVis)
library(sp)



## Read in shapefile of study area boundary(Kodiak Island)
StudyArea <- readOGR(dsn = "Data/GIS/KodiakBound/kodiak_island.shp")



## Read in the elevation, slope, aspect and
## VRM (vector ruggedness measure) rasters. Calculate elev and slope squared.
elev <- raster("Data/GIS/Elevation/dem.tif")
elev@data@names <- "Elev"
elev2 <- elev^2
elev2@data@names <- "Elev2"
slope <- raster("Data/GIS/Slope/slope.tif")
slope@data@names <- "Slope"
slope2 <- slope^2
slope2@data@names <- "Slope2"
aspect <- raster("Data/GIS/Aspect/aspect.tif")
aspect@data@names <- "Aspect"
vrm <- raster("Data/GIS/VRM/vrm.tif")

# Calculate AspectCosine (Cushman and Wallin 2002) -- A WORK IN PROGRESS
# inverse cosine of the angle - 35 degrees
values(aspect)[values(aspect) < 0] <- NA
aspectCos <- (pi * values(raster))/180

# Standardize elev^2, slope, and vrm around zero +- 1 SD. Export as grid file.
require(raster)

elevS <- (elev - cellStats(elev, stat = "mean"))/cellStats(elev, stat = "sd")
elevS@data@names <- "elevS"
writeRaster(x = elevS, filename = "Data/GIS/Elevation/elevS.grd",
            format = "raster", progress = "text", overwrite = T)

elev2S <- (elev2 - cellStats(elev2, stat = "mean"))/cellStats(elev2, stat = "sd")
elev2S@data@names <- "elevS2"
writeRaster(x = elev2S, filename = "Data/GIS/Elevation/elev2S.grd",
```

```
            format = "raster", progress = "text", overwrite = T)

slopeS <- (slope - cellStats(slope, stat = "mean"))/cellStats(slope, stat = "sd")
slopeS@data@names <- "slopeS"
writeRaster(x = slopeS, filename = "Data/GIS/Slope/slopeS.grd",
            format = "raster", progress = "text", overwrite = T)

slope2S <- (slope2 - cellStats(slope2, stat = "mean"))/cellStats(slope2, stat = "sd")
slope2S@data@names <- "slopeS2"
writeRaster(x = slope2S, filename = "Data/GIS/Slope/slope2S.grd",
            format = "raster", progress = "text", overwrite = T)

vrmS <- (vrm - cellStats(vrm, stat = "mean"))/cellStats(vrm, stat = "sd")
vrmS@data@names <- "vrmS"
writeRaster(x = vrmS, filename = "Data/GIS/VRM/vrmS.grd",
            format = "raster", progress = "text", overwrite = T)




## Read in land cover classification raster with 7 classifications:
lcc <- raster("Data/GIS/LandCover/lccNew1.tif")

# Look at it:
# plot(lcc)  # looks good

# # Define the classifications as categories for visual:
# rat <- levels(lcc)[[1]]
# rat[["landcover"]] <- c("Forest","Shrubs", "Tundra/Heath","Meadow",
#                         "Water", "Snow/Water", "Rock")
# levels(lcc) <- rat
# rm(rat)

# Make ID a factor level in the lcc raster:
lcc@data@attributes[[1]]$ID <- as.factor(lcc@data@attributes[[1]]$ID)

# Set lcc extent to match elevation raster extent:
ex <- extent(elev)
lcc <- crop(lcc, ex)

# Resample lcc ("nearest neighbor" method) to elev:
lcc <- resample(lcc, elev, "ngb")  # TAKES A WHILE! (~5-10 min)

# Export the new lcc raster as a grid file:
writeRaster(x = lcc, filename = "Data/GIS/LandCover/lcc.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Plot the lcc:
mycol <- c("green", "yellow", "light green", "brown", "blue", "white", "grey")
plot(lcc, legend = FALSE, col = mycol)
```

```
legend(x ='bottomright',
       legend = c("Forest","Shrubs", "Tundra/Heath","Meadow",
                  "Water", "Snow/Water", "Rock"),
       fill = mycol)
plot(StudyArea, border = "grey", add = T)  # Add the boundary of Kodiak Island
rm(mycol)



## Create binomial rasters of each habitat class from the lcc raster,
## based on the mean value within 50m radius of each pixel cell.

# First, need to define focal weight (50 m) for moving window (focal()):
require(raster)
fw <- focalWeight(lcc, 50, "circle")

# Then, create the rasters:

# Forest:
lcc1 <- lcc
lcc1[lcc1 > 1] <- 0
lcc1 <- focal(lcc1, w = fw, fun = "mean", na.rm = T) # mean value within "fw"
lcc1@data@names <- "forest"
writeRaster(x = lcc1, filename = "Data/GIS/LandCover/Forest.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Shrub:
lcc2 <- lcc
lcc2[lcc2 != 2] <- 0
lcc2[lcc == 2] <- 1
lcc2 <- focal(lcc2, w = fw, fun = "mean", na.rm = T)
lcc2@data@names <- "shrub"
writeRaster(x = lcc2, filename = "Data/GIS/LandCover/Shrub.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Tundra/Heath:
lcc3 <- lcc
lcc3[lcc3 != 3] <- 0
lcc3[lcc3 == 3] <- 1
lcc3 <- focal(lcc3, w = fw, fun = "mean", na.rm = T)
lcc3@data@names <- "tundraHeath"
writeRaster(x = lcc3, filename = "Data/GIS/LandCover/TundraHeath.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Meadow:
lcc4 <- lcc
```

```
lcc4[lcc4 != 4] <- 0
lcc4[lcc4 == 4] <- 1
lcc4 <- focal(lcc4, w = fw, fun = "mean", na.rm = T)
lcc4@data@names <- "meadow"
writeRaster(x = lcc4, filename = "Data/GIS/LandCover/Meadow.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Water:
lcc5 <- lcc
lcc5[lcc5 != 5] <- 0
lcc5[lcc5 == 5] <- 1
lcc5 <- focal(lcc5, w = fw, fun = "mean", na.rm = T)
lcc5@data@names <- "water"
writeRaster(x = lcc5, filename = "Data/GIS/LandCover/Water.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Snow/Water:
lcc6 <- lcc
lcc6[lcc6 != 6] <- 0
lcc6[lcc6 == 6] <- 1
lcc6 <- focal(lcc6, w = fw, fun = "mean", na.rm = T)
lcc6@data@names <- "snowWater"
writeRaster(x = lcc6, filename = "Data/GIS/LandCover/SnowWater.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Rock:
lcc7 <- lcc
lcc7[lcc7 != 7] <- 0
lcc7[lcc7 == 7] <- 1
lcc7 <- focal(lcc7, w = fw, fun = "mean", na.rm = T)
lcc7@data@names <- "rock"
writeRaster(x = lcc7, filename = "Data/GIS/LandCover/Rock.grd",
            format = "raster",
            progress = "text",
            overwrite = T)

# Clean up
rm(lcc1, lcc2, lcc3, lcc4, lcc5, lcc6, lcc7)



## Open the final spatial covariate raster layers:
elevS <- raster("Data/GIS/Elevation/elevS.grd")
elev2S <- raster("Data/GIS/Elevation/elev2S.grd")
slopeS <- raster("Data/GIS/Slope/slopeS.grd")
slope2S <- raster("Data/GIS/Slope/slope2S.grd")
```

```r
vrmS <- raster("Data/GIS/VRM/vrmS.grd")
forest <- raster("Data/GIS/LandCover/Forest.grd")
shrub <- raster("Data/GIS/LandCover/Shrub.grd")
tundraHeath <- raster("Data/GIS/LandCover/TundraHeath.grd")
meadow <- raster("Data/GIS/LandCover/Meadow.grd")
water <- raster("Data/GIS/LandCover/Water.grd")
snowWater <- raster("Data/GIS/LandCover/SnowWater.grd")
rock <- raster("Data/GIS/LandCover/Rock.grd")



## Create a raster stack of all the spatial covariates:
require(rasterVis)
r <- stack(elevS, elev2S, slopeS, slope2S, vrmS, forest, shrub, tundraHeath, meadow,
           water, snowWater, rock)

# Save the final stacked raster as a grid:
writeRaster(r, "./Data/GIS/RasterStack/Covar.grd",
            format = "raster",
            options = c("INTERLEAVE=BAND"),
            progress = "text",
            prj = T,
            overwrite = T)

# Import the stacked raster containing the covariates:
Covar <- stack("./Data/GIS/RasterStack/Covar.grd")




## Save a plot of the final raster stack:
pdf("Plots/Maps/Covariates.pdf", width = 7, height = 4, title = "Covariates")
plot(Covar)
dev.off()

# Clean up:
rm(elevS, elev2S, slopeS, slope2S, vrmS, forest, shrub, tundraHeath, meadow, water,
   snowWater, rock, elev, elev2, aspect, aspect30,aspectCos, lcc, lccJunk, ned, slope,
   slope2, vrm)
# Remove temp files:
removeTmpFiles(h = 0)

# Done. Free up cores:
EndCluster()
```

R code for extracting covariate information from rasters to used and available points

```
### STEP 2: Define "available" points at the kernel home range scale

## Create a 99% kernel HR from all "used" fixes
require(adehabitatHR)

Kernel <- kernelUD(dfSp, h = "href")
KernelVert <- getverticeshr(Kernel, percent = 99, unin = "m", unout = "km2")
crs(dfSp) <- crs(StudyArea)
# Crop the kernel to Kodiak Island (i.e., no ocean)
KernelVert <- crop(KernelVert, StudyArea)

## Generate a random sample of "available" points within the kernel:
require(sp)
AvailS <- spsample(KernelVert, type = "random", n = length(df$CollarID)*10)

# Take a look at a plot of available points:
plot(StudyArea)
plot(KernelVert, add = T)
plot(AvailS, add = T)

# Reproject available points to WGS84:
AvailS <- spTransform(AvailS, CRS = CRS("+proj=longlat +datum=WGS84"))

# Create a df of available points:
Avail <- data.frame(Response = 0, Lat = AvailS@coords[,2],
                    Long = AvailS@coords[,1])
Avail$CollarID <- rep(df$CollarID, 10)  # Add CollarID

## Bind used and available dfs together:
common_cols <- intersect(colnames(df), colnames(Avail))
dfRSF <- rbind(subset(df[common_cols]),
               subset(Avail[common_cols]))
rm(common_cols)
View(dfRSF)

dfRSFs <- SpatialPointsDataFrame(dfRSF[ ,2:1], dfRSF,
                                 proj4string = CRS("+proj=longlat +datum=WGS84"))
dfRSFs <- spTransform(dfRSFs, CRS("+proj=utm +zone=5 +datum=NAD83"))




### Extract spatial covariate raster values to used and available points:

# Required packages
require(rgeos)

## Import the raster stack:
Covar <- stack("./Data/GIS/RasterStack/Covar.grd")
```

```r
# Extract the covariate data:
dfTemp <- extract(Covar, dfRSFs[1:2],
                  method = "simple",
                  df = T)

# Bind the extracted data to the response and lat/long df:
dfRSF <- cbind(dfRSF, dfTemp)
dfRSF$Response <- as.integer(dfRSF$Response)

## Clean up
rm(dfTemp, dfRSFs, Avail, AvailS)
```

R code for RSF model selection