Data S1: Replication code for Should I shoot or should I go?

Charlotte Chang & Sarah Drohan

2018-08-27

Overview

This document contains replication code and a dataset (`SWChinaReplicationData.RData`, *Data S1*) to accompany the manuscript "Should I shoot or should I go? Simple rules for prey selection in indiscriminate harvesting systems" by Charlotte H. Chang and Sarah E. Drohan.

The central question addressed by this dataset is whether or not simple rules offered by optimal stopping problems can capture the lower bound on a trait of interest to hunters. In our case, the relevant trait is body mass. The bullet points below describe the dataset and the replication analyses performed in the script sections (a.k.a. `chunks`) below.

- Data
  - Wildlife transects: bird and mammal observations along point count transects
  - Each observation was converted to mass based on mean adult mass for each species
  - Interviews from hunters provided opportunity costs (per-period cost)
- Modeling
  - We calculate the reservation value by minimizing the function `res.function` (ε without discounting) and `res.function.dsct` (ε with δ, the discount rate).
  - We then generate the expected set of prey categories that a hunter should target under optimal foraging diet breadth modeling.

Loading data

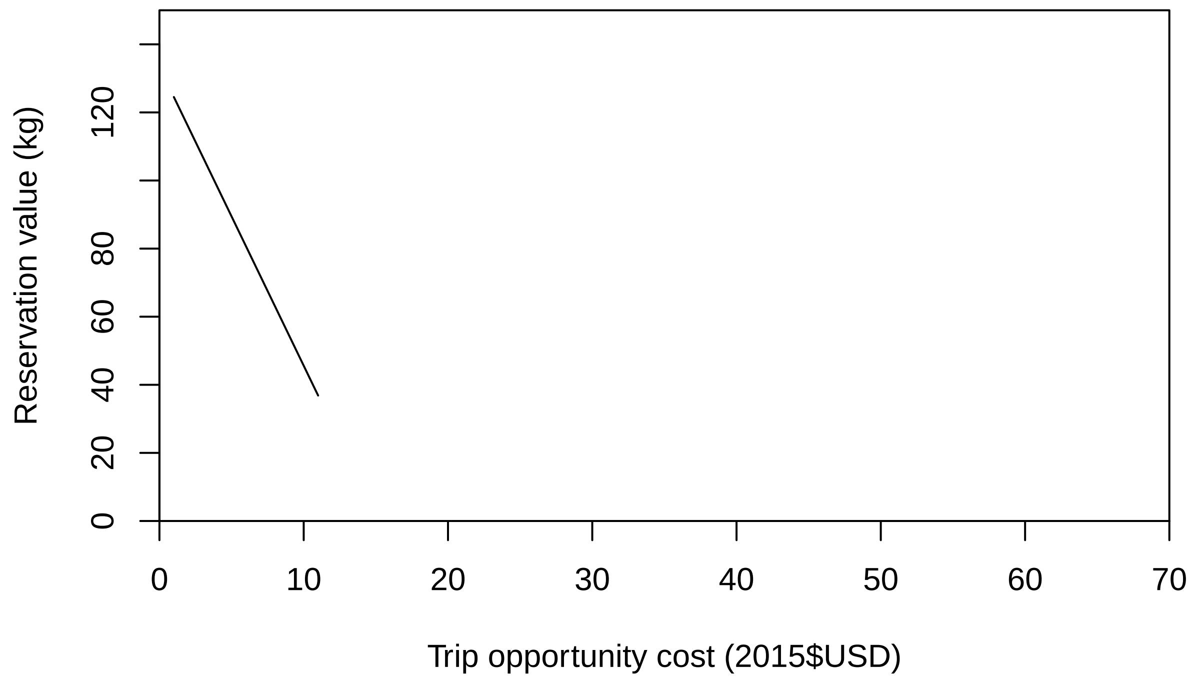
The data are provided as Supplementary Information (Data S1) to the article itself and are also available in an open-access GitHub repository (https://github.com/charlottehchang/HunterDiets).

```
### ================================================ Load
### observational data from SW China
### ================================================ File
### download from GitHub repository - you only need to do this
### once.
githubURL <- "https://github.com/charlottehchang/HunterDiets/blob/master/SWChinaReplicationData.RData?ra
download.file(githubURL, "~/Downloads/SWChinaReplicationData.RData")  # You may want to change this pat

## Loading files into workspace - Please follow this workflow.
load("~/Downloads/SWChinaReplicationData.RData")  # Ensure that you have navigated to the folder where y
# ls() # This shows you what is now contained in the
# workspace

### Calling external packages install.packages(sfsmisc) # this
### command only needs to be run once
library(sfsmisc)  # Call the package and its functions into your workspace with library
# install.packages(PMCMR) # this command also only needs to
# be run once
library(PMCMR)
```

Reservation values: thresholds for minimum trait value that will be harvested

We begin by loading the function for performing scalar optimization to find the minimum trait value (ϵ; in this case, body mass) for each hunter with a specific opportunity cost for every trip into the woods.

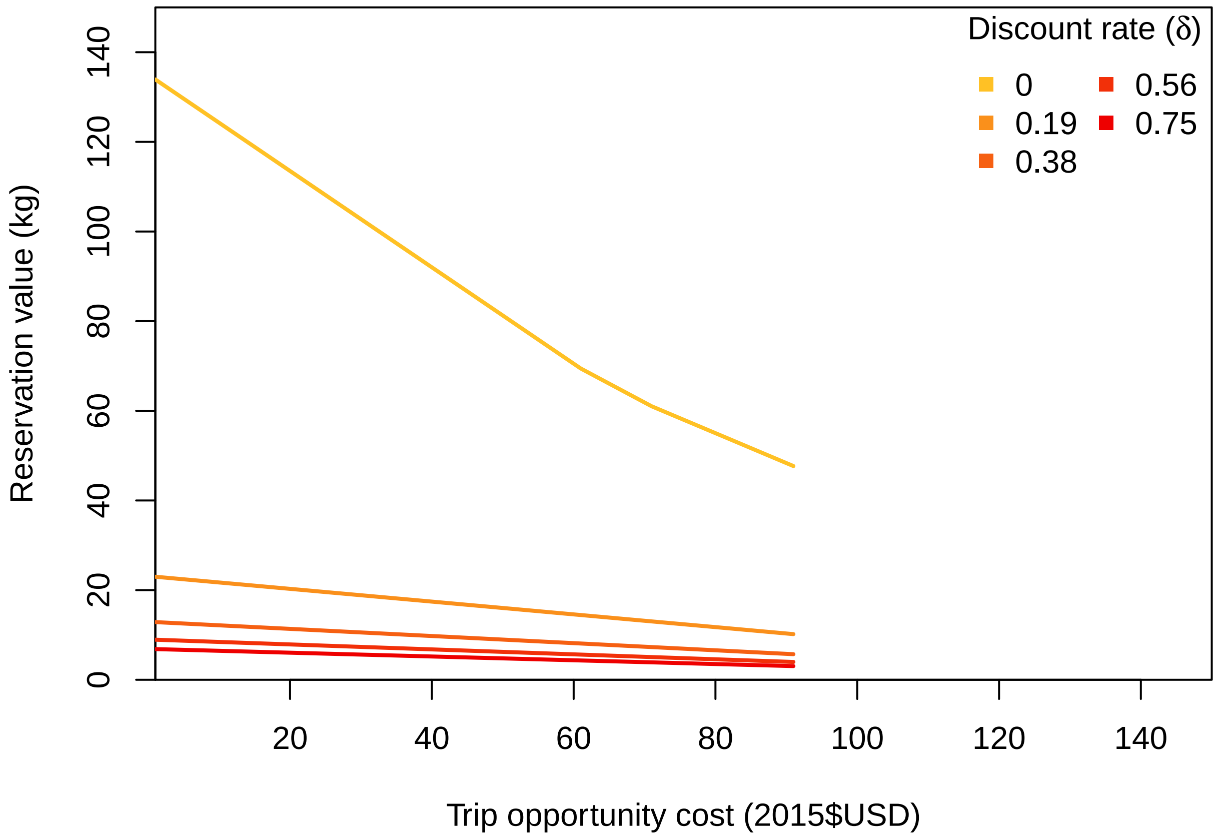
```r
### ==================================================
### Calculating example reservation values
### ==================================================

## 1: Foregone wages (USD)
costs <- seq(1, 100, by = 10)  # alternative wage cost for each hunting trip

## 2: Probability density of wildlife traits (value, and mass)
d.kgs <- density(SWChinaRepData$Mass/1000, bw = "sj", n = 2^18)
d <- density(SWChinaRepData$Value, bw = "sj", n = 2^18)  # Store density kernel estimate
## NB: To incorporate the scaling parameter, gamma, you can
## incorporate it in the calculation on the line above: d <-
## density(SWChinaRepData$Value*gamma, bw='sj', n=2^18)
## Additional parameters for epsilon calculation
mean.distro <- weighted.mean(d$x, d$y)  # Store mean of density kernel
kg.lm <- lm(d.kgs$x ~ d$x - 1)  # Get scalar to back-transform values to kilos
kg.coeff <- kg.lm$coefficients

## 3: Calculating epsilon
epsilons <- c()
for (i in 1:length(costs)) {
    # Index over diff per-period costs (opportunity cost of
    # hunting)
    if (mean.distro >= costs[i]) {
        epsilons <- append(epsilons, optimize(res.function, interval = c(min(d$x),
            max(d$x)), d = d, c = costs[i])$minimum, after = length(epsilons))  # interval, mean, and c
    } else {
        epsilons <- append(epsilons, NA, after = length(epsilons))
    }
}

## 3: Plotting
epsilons.kg <- c(epsilons * kg.coeff)  # Storing results into a matrix for plotting
plot(costs, epsilons.kg, ylab = "Reservation value (kg)", xlab = "Trip opportunity cost (2015$USD)",
    pch = 19, cex = 0.75, xlim = c(0, 70), type = "l", ylim = c(0,
        150), xaxs = "i", yaxs = "i")
```



Discounting

The `discounting` code chunk calculates the reservation value under different discount rates, otherwise holding the prey community distribution and hunter opportunity costs constant.

```r
### =================================================================
### Calculating example reservation values under discounting
### =================================================================

## 1: Discount rates (note that opportunity costs were
## specified above)
dels <- seq(0, 0.75, length.out = 5)

## 2: Calculating densities 2.1: Generate density function of
## body mass at same kernel resolution for matching economic
## value to body mass
d.kgs <- density(SWChinaRepData$Mass/1000, bw = "sj", n = 2^18)
# 2.2: Calculated scaled values
scaled.vals <- SWChinaRepData$Value * 9.75  # scale market to include amenity/bushmeat value
d.val <- density(scaled.vals, bw = "sj", n = 2^18)  # Store density kernel estimate
mean.dist <- weighted.mean(d.val$x, d.val$y)  # Store mean of density kernel
kg.lm <- lm(d.kgs$x ~ d.val$x - 1)  # Improved LM of density x (kgs) against x (USD)
kg.coeff <- kg.lm$coefficients

## 3: Calculating epsilon under different discount rates
epsilon.del <- c()
for (i in 1:length(dels)) {
    del <- dels[i]
    for (j in 1:length(costs)) {
        opp.cost <- costs[j]
        if (mean.dist >= opp.cost) {
            epsilon.del <- append(epsilon.del, optimize(res.function.dsct,
                interval = c(min(d.val$x), max(d.val$x)), d = d.val,
                c = opp.cost, del = del)$minimum, after = length(epsilon.del))
        } else {
            epsilon.del <- append(epsilon.del, NA, after = length(epsilon.del))
        }
    }
}

## 4: Preparing results
res.dsct.kgs <- kg.coeff * epsilon.del  # convert results to mass
res.dels.mat <- matrix(res.dsct.kgs, nrow = length(dels), ncol = length(costs),
    byrow = T)

## 5: Plotting
par(mar = c(4, 4, 1, 1))
col.ramp <- colorRampPalette(c("goldenrod1", "red2"))(5)
plot(costs, res.dels.mat[1, ], ylab = "Reservation value (kg)",
    xlab = "Trip opportunity cost (2015$USD)", pch = 19, cex = 0.75,
    xlim = c(1, 150), type = "n", ylim = c(0, 150), xaxs = "i",
    yaxs = "i")
for (i in 1:nrow(res.dels.mat)) {
    lines(costs, res.dels.mat[i, ], col = col.ramp[i], lwd = 2)
}
legend("topright", legend = round(dels, 2), bty = "n", col = col.ramp,
    pch = 15, ncol = 2, title = expression(paste("Discount rate (",
        delta, ")")))
```



Optimal foraging theory

The code block below calculates the optimal diet set. By definition, that is those species where

$$\frac{R_n}{h_n} \geq \frac{\sum_{i=1}^{n-1} R_i \lambda_i}{1 + \sum_{i=1}^{n-1} T_i}$$

and $T_i = \lambda_i h_i$.

```
###============================================================================
### Parameters for Optimal Foraging Theory
###============================================================================
    # 1: Total time for each group
tot.time <- (250*12)/60 # Hours for all point counts
tot.time.mammal <- (250/7)*4 # Hours across all transects

    # 2: Encounter rates
lambdas <- table(SWChinaRepData$Lab)[c('Boar','Muntjac','Pheasants','Frugivores','LargePasserines','Smal

    # 3: Masses by group
gp.kgs <- tapply(SWChinaRepData$Mass/1000,SWChinaRepData$Lab, mean)[c('Boar','Muntjac','Pheasants','Frug

    # 4: Handling times
          # Sole sources available are Koster (2010) with rifle hunter times, Table 2; and Taal Levi EE
handling.times <- c(boar = mean(c(0.7, 0.37)), # white-lipped peccary, collared peccary
                    muntjac = 0.13, # red brocket deer
                    pheasant = 0.08, # great tinamou, great currasow, plain chachalaca, little tinamou
                    frugivores = 0.05, # chestnut-mandibled toucan, parrots, crested guan
                    large.passerines = 0.05,
                    small.passerines = 0.05)

###============================================================================
### OFT diet breadth calculation
###============================================================================

# Optimal foraging reward/time
rewards.kg <- tapply(SWChinaRepData$Value,SWChinaRepData$Lab, mean)[c('Boar','Muntjac','Pheasants','Frug
raw.reward <- rewards.kg/handling.times # per-prey reward (R_i/h_i)
  # Generate OFT mean for 1..n-1 prey
ord <- order(raw.reward, decreasing=T) # get index in descending order
OFT.line <- c(raw.reward[ord[1]])
for (i in 1:5) {
  OFT.line <- append(OFT.line,sum(lambdas[ord[1:i]]*rewards.kg[ord[1:i]])/(1+ sum(handling.times[ord[1:i
}

### Diet Breadth Calculation
diet <- which(OFT.line <= raw.reward[ord])
diet
```

```
## Muntjac    Boar
##       1       2
```

Bag Back-casting

Below, we use example data from Xishuangbanna to illustrate how to apply bag back-casting. Bag back-casting is performed by comparing simulated bags from baseline wildlife communities to observed bag data from a site where the status of the exploited community is unknown. We describe how to generate simulated bags and compare them against observed catch data below.

Data and methods elements for simulated bags 1. A baseline distribution of wildlife community traits + These data were obtained from a study in the core zone of Xishuangbanna National Nature Reserve as described in Appendix S2. 2. Information on hunter opportunity costs. We use a single static cost but this method could be applied with a range of costs. 3. With these opportunity costs, one uses the baseline data, hunter opportunity costs, and Equation 1 in the main text to identify the diet threshold for individual hunters. + Alternatively, one could use the optimal foraging diet choice model to identify a diet threshold, but this would require reconfiguring one's wildlife data to estimate encounter rates, and would necessitate information on handling times for hunters processing harvested prey. 4. Subsequently, one can simulate bags from the baseline site by truncating the baseline data to all values $\geq$ the dietary threshold and bootstrap sampling items.

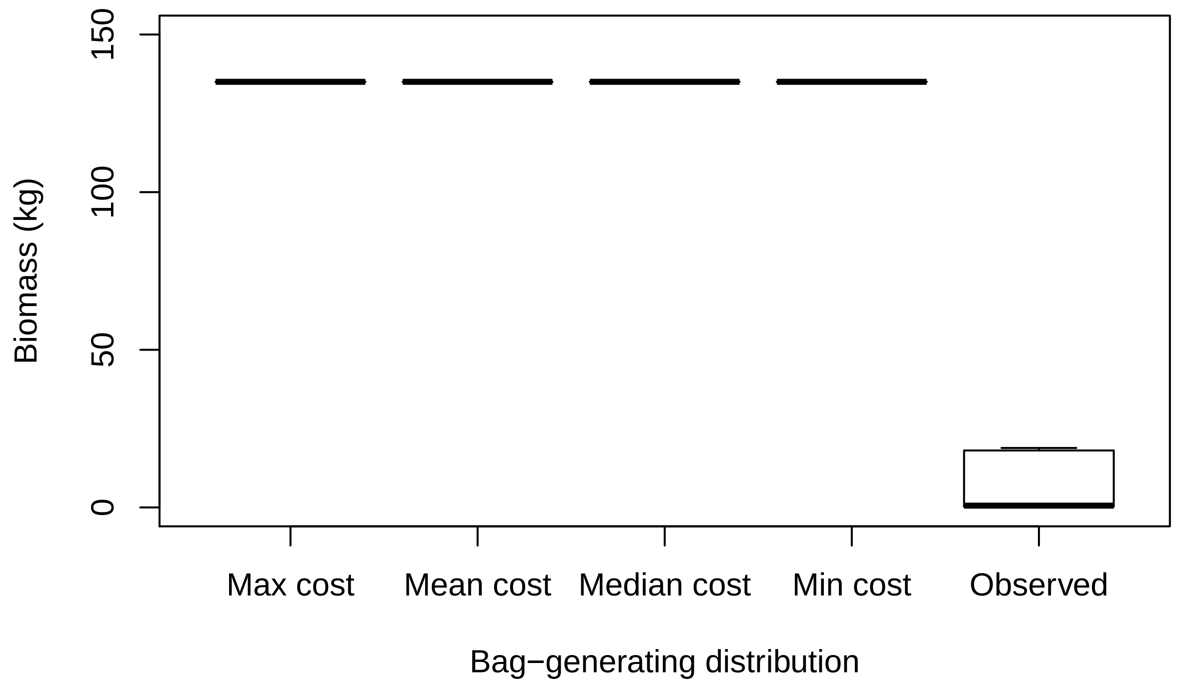
Proceeding to comparison of baseline bags against observed bags 5. One method to compare the bag data against the simulated baseline bags is to perform a comparison of means across all the bags and then run posthoc tests.

```
### ==========================================================================
### Bag back-casting: Part 1 - Null (historical) trait
### distribution
### ==========================================================================

# This is provided in the data.frame xsbn.null from the
# replication .rda file
str(xsbn.null)
```

```
## 'data.frame':    707 obs. of  2 variables:
##  $ Mass : num  7603 7603 7603 7603 7603 ...
##  $ Value: num  23.6 23.6 23.6 23.6 23.6 ...
```

```
### ==========================================================================
### Bag back-casting: Part 2 - Dietary threshold
### ==========================================================================

d <- density(xsbn.null$Value, bw = "sj", n = 2^18)
vals.lm <- 0.00310571

# Mean = 24.189, Median = 16.32
mean.cost = 24.19
median.cost = 16.32
min.cost = 2
max.cost = 68

epsilon.null.mean <- optimize(res.function, interval = c(min(d$x),
    max(d$x)), d = d, c = mean.cost)$minimum

epsilon.null.median <- optimize(res.function, interval = c(min(d$x),
    max(d$x)), d = d, c = median.cost)$minimum

epsilon.null.min <- optimize(res.function, interval = c(min(d$x),
    max(d$x)), d = d, c = min.cost)$minimum
```

```r
epsilon.null.max <- optimize(res.function, interval = c(min(d$x),
    max(d$x)), d = d, c = max.cost)$minimum

epsilon.kgs <- c(mean = epsilon.null.mean * 1/vals.lm * 1/1000,
    median = epsilon.null.median * 1/vals.lm * 1/1000, min = epsilon.null.min *
        1/vals.lm * 1/1000, max = epsilon.null.max * 1/vals.lm *
        1/1000)

### ==========================================================================
### Bag back-casting: Part 3 - Bootstrapping bags
### ==========================================================================

epsilon.g <- epsilon.kgs * 1000

set.seed(100)
bag.boot <- c(sample(xsbn.null$Mass[xsbn.null$Mass >= epsilon.g[1]],
    1000, replace = TRUE), sample(xsbn.null$Mass[xsbn.null$Mass >=
    epsilon.g[2]], 1000, replace = TRUE), sample(xsbn.null$Mass[xsbn.null$Mass >=
    epsilon.g[3]], 1000, replace = TRUE), sample(xsbn.null$Mass[xsbn.null$Mass >=
    epsilon.g[4]], 1000, replace = TRUE))

bag.df <- data.frame(bags = c(bag.boot/1000, catch.biomass[catch.biomass >
    0]/1000), lab = c(rep("Mean cost", 1000), rep("Median cost",
    1000), rep("Min cost", 1000), rep("Max cost", 1000), rep("Observed",
    length(catch.biomass[catch.biomass > 0]))))

### ==========================================================================
### Bag back-casting: Part 4 - Visualizing simulated bags
### ==========================================================================

boxplot(bags ~ lab, data = bag.df, xlab = "Bag-generating distribution",
    ylab = "Biomass (kg)", ylim = c(0, 150), outline = F)
```



```
### ==========================================================================
### Bag back-casting: Part 5 - Comparing bags against null bag
### ==========================================================================

### Comparison of means
bags.test <- kruskal.test(bags ~ lab, data = bag.df)
bags.test
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  bags by lab
## Kruskal-Wallis chi-squared = 1701.1, df = 4, p-value < 2.2e-16
```

```
bags.posthoc <- PMCMR::posthoc.kruskal.nemenyi.test(bags ~ lab,
    data = bag.df)
```

```
## Warning in posthoc.kruskal.nemenyi.test.default(c(135, 135, 135, 135,
## 135, : Ties are present, p-values are not corrected.
```

```
bags.posthoc
```

```
##
##  Pairwise comparisons using Tukey and Kramer (Nemenyi) test
##                    with Tukey-Dist approximation for independent samples
##
## data:  bags by lab
##
##             Max cost Mean cost Median cost Min cost
## Mean cost   1        -         -           -
## Median cost 1        1         -           -
## Min cost    1        1         1           -
## Observed    4.1e-14  4.1e-14   4.1e-14     4.1e-14
##
```

```
## P value adjustment method: none
```