# Comparative analysis of methods for inferring successful foraging areas from Argos and GPS tracking data

Anne-Cécile Dragon[1,2,4], Avner Bar-Hen[2], Pascal Monestiez[3], and Christophe Guinet[1]

[1]CEBC-CNRS, France

[2]MAP5-CNRS, France

[3]BioSP-INRA, France

[4]acdlod@locean-ipsl.upmc.fr

December 7, 2011

# Supplementary Material

## Drift Dives detection and Assessment of the seals' body conditions

*R* code to detect drift dives from TDR data of Southern Elephant seals

```
setwd("")
library(caTools)

TDRdata <- read.table(file="TDRdata.txt",sep="\t",header=T,
    colClasses=c("POSIXct","numeric"))
# TDRdata.txt contains 2 columns:
# "Date.Time" (POSIXct format) and "Depth" (in meters)

n <- nrow(TDRdata)

# calculation of the vertical speed
    numerator <- diff(TDRdata$Depth)
    denominator <- diff(as.numeric(TDRdata$Date.Time))
    Vertical.Speed <- c(0, -numerator / denominator)

# smoothing of the vertical speed over a 10 seconds window
# (data temporal resolution = 2 seconds)
    TDRdata$Smooth.Vert.Speed <- runmean(Vertical.Speed,5)

# creation of the drift vectors:
# one for the negative drift phases and the other for the positive ones
    TDRdata$drift.N <- TDRdata$drift.P <- logical(n)

# parameters of drift detection (to be adjusted according to the data resolution)

# "alpha" corresponds to the drift detection threshold (in m/s):
```

```
# under this vertical speed, the animal is thought to swim passively
    alpha <- 0.6

# "beta" is the minimal duration for a passive phase to be considered as a drift phase:
# an animal is drifting only if its passive swimming lasted more than beta minutes
    beta <- 90  # 3min = 180 sec = 90 data points

# "gamma" corresponds to the maximal duration tolerated outside the threshold:
# if the animal swims over the threshold during more than gamma seconds,
# the drift phase is considered to be over
    gamma <- 4 # 4 data points = 8 seconds

# "delta" is the homogeneity parameter:
# the animal is considered drifting when the vertical speed variance
# over the passive phase is inferior to delta
    delta <- 0.05

# Detection of the Negative Drift Phases
    Neg <- vector("numeric",length=n)
    sel <- which((TDRdata$Smooth.Vert.Speed < 0) & (TDRdata$Smooth.Vert.Speed > -alpha))
    Neg[sel] <- TRUE
    diff.N <- c(0,diff(Neg))
    # location of the beginning of the negative passive phases:
    start.N <- which(diff.N > 0)
    if(start.N[length(start.N)]==n)){
    start.N <- start.N[-length(start.N)] }
    # location of the end of the negative passive phases:
    end.N <- which(diff.N < 0)

    # passive phases separeted by less than gamma seconds are pooled together:
    dum <- start.N[2:length(start.N)] - end.N[1:(length(start.N)-1)]
```

```
    sel.short.seq <- which(dum < (gamma + 1))
    start.N <- start.N[-(sel.short.seq+1)]
    end.N <- end.N[-sel.short.seq]

    # passive phases lasting less than beta minutes are discarded
    start.drift.N <- start.N[which((end.N - start.N) > beta)]
    end.drift.N <- end.N[which((end.N - start.N) > beta)]

    for (j in 1:length(start.drift.N)){
     TDRdata$drift.N[start.drift.N[j]:end.drift.N[j]] <- TRUE }

# Detection of the Positive Drift Phases
    Pos <- vector("numeric",length=n)
    sel <- which((TDRdata$Smooth.Vert.Speed > 0) & (TDRdata$Smooth.Vert.Speed < alpha))
    Pos[sel] <- TRUE
    diff.P <- c(0,diff(Pos))
    # location of the beginning of the positive passive phases:
    start.P <- which(diff.P > 0)
    if(start.P[length(start.P)] == n)){ start.P <- start.P[-length(start.P)] }
    # location of the end of the positive passive phases:
    end.P <- which(diff.P < 0)

    # passive phases separeted by less than gamma seconds are pooled together:
    dum <- start.P[2:length(start.P)] - end.P[1:(length(start.P)-1)]
    sel.short.seq <- which(dum < (gamma + 1))
    start.P <- start.P[-(sel.short.seq+1)]
    end.P <- end.P[-sel.short.seq]

    # passive phases lasting less than beta minutes are discarded
    start.drift.P <- start.P[which((end.P - start.P) > beta)]
    end.drift.P <- end.P[which((end.P - start.P) > beta)]
```

```
    for (j in 1:length(start.drift.P)){
     TDRdata$drift.P[start.drift.P[j]:end.drift.P[j]] <- TRUE }
```

# Hidden Markov Models

## Double switch with no covariate

*WinBUGS* code to fit a "double switch" Hidden Markov Model with no covariate (*lines in italics are the ones changed in the models with covariates*)

```
double.switch.model <- function(){

# priors of the transition probabilities from one behavioural mode to the other:
# behavioural mode 1: slow and tortuous displacement ( = intensive foraging)
# behavioural mode 2: high displacement speed and linear way of moving
( = migration towards the next food patch and/or extensive foraging)
```

$q[1] \sim dunif(0,1)$

$q[2] \sim dunif(0,1)$

```
    # priors for the shape parameter of Weibull distribution (for the StepLength data)
    nu.Step[2] ~ dgamma(0.01, 0.01)
    nu.Step[1] ~ dgamma(0.01, 0.01)

    # priors for the scale parameter of Weibull distribution
    lambda.Step[1] ~ dgamma(0.01, 0.01)
    lambda.Step[2] ~ dgamma(0.01, 0.01)

    # priors for the mean turning angles
    nu.Ang[1] ~ dunif(-3.14159265359, 3.14159265359)
    nu.Ang[2] ~ dunif(-3.14159265359, 3.14159265359)
```

```
# priors for the mean cosinus of the circular distribution
    lambda.Ang[2] ~ dunif(-1,1)
    lambda.Ang[1] ~ dunif(lambda.Ang[2],1) # so that lambda.Ang[1] > lambda.Ang[2]

# initialisation t=1
    for (t in 1:1) {

# probability of being in mode 1 or 2 at time t = 1
   dummy ~ dunif(0,1)
   p[1,1] <- dummy
       p[1,2] <- 1 - p[1,1]

# b.mode is the hidden behavioural variable
       b.mode[1] ~ dcat(p[1,])

# estimation of the animal steplength at time t with a Weibull distribution
       a[t] <- nu.Step[b.mode[t]]
       b[t] <- lambda.Step[b.mode[t]]
       steplength[t] ~ dweib(b[t], a[t])

# estimation of the animal turning angle, theta[t], at time t
# with a Wrapped Cauchy distribution
       ones[t] <- 1
       ones[t] ~ dbern(wc[t])
       rho[t] <- lambda.t[b.mode[t]]
       mu.Ang[t] <- nu.Ang[b.mode[t]]
       term1 <- 1 / (2*3.14159265359)
       term2 <- (1-rho[t]*rho[t]) / (1+rho[t]*rho[t]-2*rho[t]*cos(theta[t]-mu.Ang[t]))
       wc[t] <- term1 * term2 /300

```

```
    }

# N iterations: loop over the duration of the animal track
    for (t in 2:N) {

      a[t] <- nu.Step[b.mode[t]]
      b[t] <- lambda.Step[b.mode[t]]
      steplength[t] ~ dweib(b[t], a[t])
      ones[t] <- 1
      ones[t] ~ dbern(wc[t])
      term1 <- 1 / (2*3.14159265359)
      term2 <- (1-rho[t]*rho[t]) / (1+rho[t]*rho[t]-2*rho[t]*cos(theta[t]-mu.Ang[t]))
      wc[t] <- term1 * term2 /300
      rho[t] <- lambda.Ang[b.mode[t]]
      mu.Ang[t] <- nu.Ang[b.mode[t]]

      # estimation of the behavioural mode at time t:
      b.mode[t] ~ dcat(p[t,])
```

$$p[t,1] = q[b.mode[t-1]]$$

```
      p[t,2] <- 1 - p[t,1]]

    }
}
```

## Double switch with one covariate

*WinBUGS* code to fit a "double switch" Hidden Markov Model with a logit link and Sea Level Anomalies (or Bottom Time Residuals) as a covariate

`same model as the one described in the previous subsection, but the influence of the covariate for each time step (cov.data[t]) is added through a logit link:`

```
# priors for the covariate
     a0[1] ~ dnorm(0,0.001)
     a0[2] ~ dnorm(0,0.001)
     a1[1] ~ dnorm(0,0.001)
     a1[2] ~ dnorm(0,0.001)

       logit(q[t,1]) <- a0[1] + a1[1] * cov.data[t]
       logit(q[t,2]) <- a0[2] + a1[2] * cov.data[t]
       p[t,1] <- q[t,b.mode[t-1]]
       p[t,2] <- 1 - p[t,1]
```

## Double switch with environmental and behavioural covariates

*WinBUGS* code to fit a "double switch" Hidden Markov Model with a logit link and Sea Level Anomalies and Bottom-Time Residuals as covariates

```
same model as the one described in the previous subsection, but a multiple regression
with the 2 covariates (sla[t], sea level anomalies, and resBT[t], bottom time residuals)
is included in the logit link:
# priors for the covariate
     a0[1] ~ dnorm(0,0.001)
     a0[2] ~ dnorm(0,0.001)
     a1[1] ~ dnorm(0,0.001)
     a1[2] ~ dnorm(0,0.001)
     a2[1] ~ dnorm(0,0.001)
     a2[2] ~ dnorm(0,0.001)
     a3[1] ~ dnorm(0,0.001)
     a3[2] ~ dnorm(0,0.001)

logit(q[t,1]) <- a0[1] + a1[1] * sla[t] +
       a2[1] * resBT[t] + a3[1]*resBT[t]*sla[t]
```

```
logit(q[t,2]) <- a0[2] + a1[2] * sla[t] +
        a2[2] * resBT[t] + a3[2]*resBT[t]*sla[t]

        p[t,1] <- q[t,b.mode[t-1]]
        p[t,2] <- 1 - p[t,1]
```