Right Whale Health Model Elements

Rob Schick

8/7/2120

Response to Questions

We met on 31 July to discuss a few features related to modeling health in right whales. FWC researchers asked a few questions regarding missing data imputation, variance around latent estimates of health, and estimating age. Here I provide a quick attempt to address each of these.

Missing Data

While the right whale dataset is very long and covers a great number of individuals, and while the Visual Health Assessment data are a fantastic resource, there are still a lot of missing data. Since the PCOD model provides estimates of latent states of health given observations of health, we address this issue through partial imputation of missing VHA data. We do that in two steps.

Initialization

First, in the initialization routine `rightWhaleInput.r` we impute missing data for body condition and skin condition. The code starting around line 355 uses a `which()` statement to flag all of the missing entries in the two observed data matrices `hobs` and `skobs`. (The reason we only impute these was based on discussion with NEAq, where they felt they were the two most important indicators of condition, and the ones they felt most confident about w/r/t imputation.)

The main block of the initialization routine is:

```r
if(missDataImp){

  # Body Fat
  hTable <- rbind( c(5:1), c(1, 5, 10, 5, 1), c(1:5) )
  hTable[,3] <- c(1, 5, 1)
  hTable <- hTable / matrix( colSums(hTable), 3, 5 ,byrow = TRUE)
  ntab   <- ncol(hTable)

  hStates <- initH6months(hobs) # linearly interpolates on body fat scale
  hStatePrior <- makeHPrior6months(hobs, hStates, skinFlag = FALSE) # linearly interpolates on hTable s
  pregidx <- which(calves == 2 & is.na(hobs) & hStatePrior != 5)
  hStatePrior[pregidx] <- 5

  # Skin
  skTable <- rbind( c(5:1), c(1:5) )
  skTable <- skTable / matrix( colSums(skTable), 2, 5 ,byrow = TRUE)

  ntab   <- ncol(skTable)

  skin <- initH(skobs) # linearly interpolates on skin scale
  skinStatePrior <- makeHPrior(skobs, skin, skinFlag = TRUE) # linearly interpolates on hTable scale

}
```

where we set up initial values for missing data +/- 6 months around a sighting (this is in `hStates` and `skin`). These tables allow for linear interpolation when the observed health changes between sightings. Note that because body condition has an extra ordinal class, its dimensions are larger. Here are the two normalized matrices, first for body condition:

```
hTable <- rbind( c(5:1), c(1, 5, 10, 5, 1), c(1:5) )
hTable[,3] <- c(1, 5, 1)
hTable <- hTable / matrix( colSums(hTable), 3, 5 ,byrow = TRUE)
round(hTable, 2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.71 0.36 0.14 0.18 0.14
## [2,] 0.14 0.45 0.71 0.45 0.14
## [3,] 0.14 0.18 0.14 0.36 0.71
```

and then for skin:

```
skTable <- rbind( c(5:1), c(1:5) )
skTable <- skTable / matrix( colSums(skTable), 2, 5 ,byrow = TRUE)
round(skTable, 2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.83 0.67  0.5 0.33 0.17
## [2,] 0.17 0.33  0.5 0.67 0.83
```

with the matrix providing a way to stochastically interpolate for the times between observations. For example, if there's an observation in one state at one time and a different state at a new time.

We then take the observed data and initialized imputed values with the `initH6months()` function:

```
initH6months <- function(hObs){
  # In contrast to initH(), this just interpolates if the sightings are within 6 months of a missing da

  hh    <- hObs*0 + 1
  tvec  <- 1:ncol(hObs)
  hTobs <- hObs
  hp    <- hTobs
  nt    <- ncol(hObs)
  n     <- nrow(hObs)

  for(i in 1:n){
    firstObs <- min(which(is.finite(hTobs[i,])), na.rm = T)
    lastObs <- max(which(is.finite(hTobs[i,])), na.rm = T)

    for(t in 1:nt){
      if(is.finite(hTobs[i, t]))next

      if(t < firstSight[i]){next}
      if(t < firstObs & abs(firstObs - t) < 6){hp[i, t] <- hTobs[i, firstObs]} # Scenario 1

      nlast <- max(hh[i, 1:t] * tvec[1:t], na.rm = T)
      nnext <- min(hh[i, t:nt] * tvec[t:nt], na.rm = T)
      nlapsb <- t - nlast
      nlapsf <- nnext - t

      if(nlapsb <= 6 & nlapsf > 6){hp[i, t] <- hTobs[i, nlast]} # Scenario 3
      if(nlapsb > 6 & nlapsf <= 6){hp[i, t] <- hTobs[i, nnext]} # Scenario 4
      if(nlapsb > 6 & nlapsf > 6){next} # Scenario 5
      if(t > lastObs & t - lastObs < 6){hp[i, t] <- hTobs[i, lastObs]} # Scenario 6
      if(nlapsb <= 6 & nlapsf <= 6){# Scenario 2
      tlast  <- hTobs[i, nlast ]
      tnext  <- hTobs[i, nnext ]
      h0     <- hTobs[i, nlast]
      if(!is.finite(nnext))nnext <- t
      if(!is.finite(tlast))tlast <- tnext
      if(!is.finite(tnext))tnext <- tlast
      if(!is.finite(nlast)){
        nlast <- t
        h0    <- tlast
      }

      slope  <- (tnext - tlast)/(nnext - nlast)
      hp[i, t] <- h0 + slope * (t - nlast)
    }
   }
  }
  round(hp,0)
}
```

So this simply fills in the missing body condition data for the time periods 6 months before and after an observation. We now have initialized values for the **hStates** matrix prior to the Gibbs loop. Here's a snippet of before and after. The left hand column is the observed raw data, and the right hand column is the initialized data. Two things to note: 1) you can see that you only get imputed data for the 6 months (e.g. rows 3-8); and 2) you see the linear interpolation between an observation of a body condition score of 2 and 3 (e.g. rows 18-22 with the switch in classes happening between rows 20 and 21):

```
> cbind(hobs[4, 316:346], hStates[4, 316:346])
      [,1] [,2]
 [1,]   NA   NA
 [2,]   NA   NA
 [3,]   NA    2
 [4,]   NA    2
 [5,]   NA    2
 [6,]   NA    2
 [7,]   NA    2
 [8,]   NA    2
 [9,]    2    2
[10,]    2    2
[11,]    2    2
[12,]   NA    2
[13,]   NA    2
[14,]   NA    2
[15,]    2    2
[16,]    2    2
[17,]    2    2
[18,]    2    2
[19,]   NA    2
[20,]   NA    2
[21,]   NA    3
[22,]   NA    3
[23,]    3    3
[24,]    3    3
[25,]   NA    3
```

```
[26,]   NA    3
[27,]   NA    3
[28,]   NA    3
[29,]   NA    3
[30,]   NA    3
[31,]   NA   NA
```

The imputation routine is continuous, but of course the imputed values are ordinal to match with NEAq's scale.

Prior

Setting up the prior happens next with the `makeHPrior6months(hobs, hStates, skinFlag = FALSE)` function. The arguments are: `hobs` the observed data, `hStates` the observed data with the initialized imputed values, and the FLAG for whether we're imputing for skin or body (n.b. this is simply to deal with the different dimensions of the ordinal class).

If we add this prior to the above data, we can begin to understand the structre of the data:

```
> cbind(hobs[4, 316:346], hStates[4, 316:346], hStatePrior[4, 316:346])
      [,1] [,2] [,3]
 [1,]   NA   NA   NA
 [2,]   NA   NA   NA
 [3,]   NA    2    3
 [4,]   NA    2    3
 [5,]   NA    2    3
 [6,]   NA    2    3
 [7,]   NA    2    3
 [8,]   NA    2    3
 [9,]    2    2    3
[10,]    2    2    3
[11,]    2    2    3
[12,]   NA    2    3
[13,]   NA    2    3
[14,]   NA    2    3
[15,]    2    2    3
[16,]    2    2    3
[17,]    2    2    3
[18,]    2    2    3
[19,]   NA    2    3
[20,]   NA    2    3
[21,]   NA    3    5
[22,]   NA    3    5
[23,]    3    3    5
[24,]    3    3    5
[25,]   NA    3    5
[26,]   NA    3    5
[27,]   NA    3    5
[28,]   NA    3    5
[29,]   NA    3    5
[30,]   NA    3    5
[31,]   NA   NA   NA
```

As above, this simply linearly interpolates and we will use this in the Gibbs loop to impute the missing data. We use the prior to sample the imputed data in the `updateHealth()` function:

```r
# Body Fat
hProb <- predProb(bh, cnow) * t( hTable[, hStatePrior[wa, t]] )
hProb[hProb < 0] <- 0
hProb <- hProb / matrix(rowSums(hProb), na, 3)
hh <- rowSums(myrmultinom(1, hProb) * matrix(c(1:3), na, 3, byrow = T))
hsNew[wa, t] <- hh

hStates[missH] <- hsNew[missH]
```

Variance around latent estimates of health

Ok, next question was around how we propose the latent estimates of health. First, health is proposed in the `propH()` function, which is called in the `updateHealth()` function. The inputs for the `propH` function are 1) the current estimated values of health (`cnow`) for the animals imputed to be alive at each time step (`health[wa, t]`), and 2) the `hoffset` scalar which operates to limit the proposal values.

Specifically, we propose new health values from a truncated normal distribution, where '`hoffset` serves as the upper and lower bounds on the proposal:

```r
propH <- function(cnow, hoffset = 5){

  clo <- cnow - hoffset
  clo[clo < 0] <- 0
  chi <- cnow + hoffset
  chi[chi > 100] <- 100

  tnorm(length(cnow), clo, chi, cnow, rexp(length(cnow), 1/10))

}
```

Next in terms of summarizing the uncertainty in the estimates, this is done as follows after the Gibbs loop is complete. In the Gibbs loop, current health and the standard deviation is stored as:

```r
sumh <- sumh + hh
sumh2 <- sumh2 + hh^2
```

Where `hh` is an `n` by `nt` matrix containing the current estimate of health across animals and time.

First since through the loop we are storing the moments, we use the number of Gibbs steps to create an `n` by `nt` matrix of mean health values:

```r
healthmean <- sumh / ngg
```

The variance is then calculated as:

```r
hvar <- sumh2 / ngg - healthmean ^ 2

healthsd   <- sqrt(hvar)
```

In plotting the time series of individual health, we then use `healthmean` and `healthsd`.

Age

Lastly for now, we don't estimate age, but instead tie survival to age, as a function of when the animal was first seen, the current time period, and the estimated or observed date of death. If the animal is a juvenile, we include that as a separate covariate in the parameters for health. Also note that we could never get confergence on the parameter for health as a function of juvenile age, so for a while now we haven't been trying to estimate it.

See the `healthStateSpacePars()` function for how these parameters are proposed and accepted.