Using predictp

David Borchers

2024-10-08

## Overview

This package combines estimates of detection probability given depth (obtained from an at-sea aerial survey experiment) and estimates of the proportion of time that animals spend at each depth (obtained from ARGOS tags and V-tags), to predict the probability of detecting an animal when we don’t know its depth.

We refer to the model for detection probability given depth as the “visibility model” and the model for proportion of time at depths as the “tag model”.

(To cut to the chase, jump to the section **How to Predict Detection Probability** below.)

### The Visibility Model

Detection probability given depth is modelled as a function of the variables below. The model itself is a GAM and the list below gives the covariates that it requires in order to predict detection probability.

* pos: The position the observer (camera) is scanning. This is a factor with possible values “Centre”, “Near” or “Far”. NOTE that pos need not be specified, and if it is not, the resulting detection probability is the average predicted probability across the positions.
* secchi: A measure of the maximum depth that a Secchi disk can be seen. This is a continuous variable. The data from experiment used to construct the GAM model used secchi ranging from 1.5 to 7.5.
* cloud: A measure of cloud cover, with higher numbers being more cloud.The data from experiment used to construct the GAM model used cloud ranging from 0 to 8.
* hdglare: The HiDef Ltd measure of glare. This is a factor with levels “1”, “2” or “3”.

The tag model is in the gam object called vismodel, which can be accessed after loading the library as follows

library(predictp)  
data("vismodel")

You can look at it like this

vismodel  
#>   
#> Family: binomial   
#> Link function: logit   
#>   
#> Formula:  
#> seen ~ pos + s(secchi, k = 4) + s(SS, k = 4) + s(cloud, depth,   
#> k = 8) + hdglare  
#>   
#> Estimated degrees of freedom:  
#> 2.23 2.13 6.11 total = 15.47   
#>   
#> UBRE score: -0.09648188

To predict, it requires a data frame with columns as specified above. There are two example data frames in the package, called eg\_viscov and eg\_viscov\_withpos. They are identical, except that the former does not have a “pos” column:

data("eg\_viscov")  
data("eg\_viscov\_withpos")  
eg\_viscov  
#> secchi SS cloud hdglare  
#> 1 8 0 0 1  
#> 2 1 0 2 2  
#> 3 8 1 4 3  
#> 4 1 1 4 1  
#> 5 8 2 2 2  
#> 6 1 2 0 3  
eg\_viscov\_withpos  
#> pos secchi SS cloud hdglare  
#> 1 Centre 8 0 0 1  
#> 2 Centre 1 0 2 2  
#> 3 Near 8 1 4 3  
#> 4 Near 1 1 4 1  
#> 5 Far 8 2 2 2  
#> 6 Far 1 2 0 3

### The Tag Model

The tag model has two components, a GAM model constructed from ARGOS tag data, and the observed proportion of times that animals spend at, or shallower than, each one of 7 depths from 0m to 3m at 0.5m depth intervals (from Tielman et al., 2013).

These proportions of times are contained in the object cdfdataframe in the package, which you can look at like this:

data("cdfdataframe")  
cdfdataframe  
#> depth cdf  
#> 1 0.0 0.124  
#> 2 0.5 0.321  
#> 3 1.0 0.421  
#> 4 1.5 0.462  
#> 5 2.0 0.490  
#> 6 2.5 0.510  
#> 7 3.0 0.527

The GAM model is requires the following covariates:

* periodfac: A factor indicating the period of in the day, with levels “0”, “1”, “2”, “3”.
* diy: An integer giving the day of the year, with January 1st as 1.

This model is in the gam object called tagmodel, which can be accessed as follows

data("tagmodel")

You can look at it like this

tagmodel  
#>   
#> Family: Beta regression(9.156)   
#> Link function: logit   
#>   
#> Formula:  
#> tad ~ diy + periodfac + s(diy, bs = "cc") + s(id, bs = "re") +   
#> s(id, periodfac, bs = "re") + s(id, diy, bs = "re")  
#>   
#> Estimated degrees of freedom:  
#> 6.59 30.69 70.86 26.56 total = 139.71   
#>   
#> REML score: -1924.09

It requires a data frame with columns as specified above. There is an example data frame in the package, called eg\_tagcov:

data("eg\_tagcov")  
eg\_tagcov  
#> periodfac diy  
#> 1 0 110  
#> 2 3 50  
#> 3 2 20  
#> 4 1 120  
#> 5 3 220  
#> 6 0 330

### How to Predict Detection Probability

The key function for predicting detection probability is the function psee. To get predicted detection probabilities you just pass it a visibility covariate data frame, a tag covariate data frame with the same number of rows, vismodel and tagmodel, in that order. For example, using the visibility covariate data frame eg\_viscov and the tag covariate data frame eg\_tagcov:

p = psee(eg\_viscov, eg\_tagcov, vismodel, tagmodel)

This gives us these estimates of detection probability: 0.43, 0.132, 0.177, 0.201, 0.278, 0.135. (The data frame cdfdataframe is used inside the function psee.)

## What Happens Under the Hood

The visibility model is a model for , where is depth and is a vector of covariates for the visibility model, as specified above (e.g. a row of the data frame eg\_viscov).

The tag model and the information on proportion of times at depth that is contained in cdfdataframe together comprise a model for the cumulative distribution function of depths of animals, , where is a vector of covariates for the tag model, as specified above (e.g., a row of the data frame tagcov). Or equivalently the comprise a model for the probability density function of animal depths, .

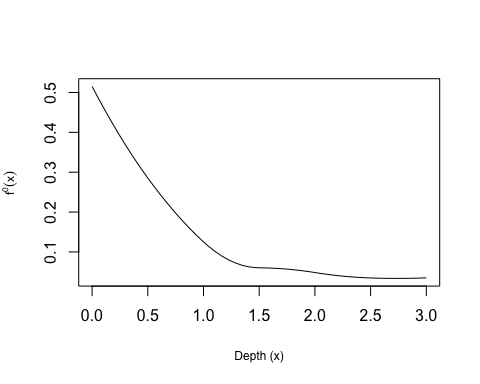
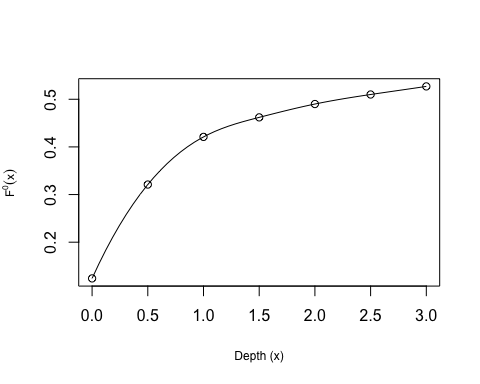
All that the function psee does, is calculate the integral , which you can think of as a weighted average detection probability across all depths between 0 and 3, with weighs proportional to the time animals spend at each depth.

### Constructing

The ARGOS tag data is such that we can estimate from it only the probability that an animal is within 2m of the surface, i.e., . That is, we can only estimate for the single value from these data, not the whole function .

The V-tag data gives us higher resoltion data in the depth () dimension. We can get an estimate of the relative proportion of time animals spend shallower than any depth (up to depth 3m) by smoothing through the V-tag proportions contained in cdfdataframe. Let’s call this curve and the corresponding probability density function . The figure below shows (left) and (right), while the dots in the left plot show the data in cdfdataframe.

# Make a smooth interpolating spline function for the CDF  
depthcdf = splinefun(cdfdataframe$depth,cdfdataframe$cdf)  
# ... and plot it:  
x = seq(0,3,length=50)  
plot(x,depthcdf(x),type="l",xlab="Depth (x)",ylab=expression({F^0}(x)),cex.lab=0.75)  
points(cdfdataframe$depth,cdfdataframe$cdf)  
plot(x,depthcdf(x,deriv=1),type="l",xlab="Depth (x)",ylab=expression({f^0}(x)),cex.lab=0.75)



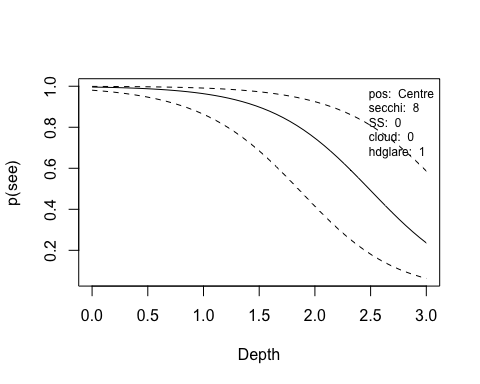
We can use the V-tag data to extend the model for obtained from the ARGOS tag data, to all if we assume that the *relative* proportion of time animals spend at each depth stays constant, and that all the covariates do is to adjust the absolute proportion of time they spend at each depth. In this case, , from which we get the probability density function that we need for the integral above, in order to calculate .

## Some plotting functions

# Plotting the visibility model

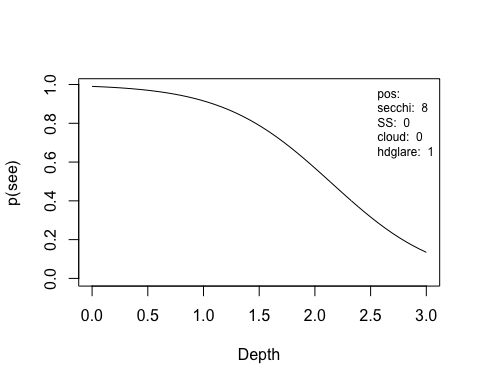
The predictp package contains a function to plot the visibility model, , which is called plotdetfn. You pass it the visibility model (vismodel) and a single row from a visibility covriate data frame (like eg\_viscov\_withpos or eg\_viscov, for example). When the data frame contains a column pos 95% confidence intervals (CIs) are plotted (dotted lines), like this

plotdetfn(vismodel,eg\_viscov\_withpos[1,])



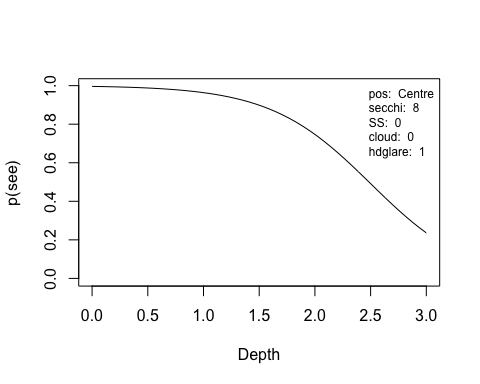
but when it does not, you get a warning and no 95% CI lines - because this has not yet been implemented in the package.

plotdetfn(vismodel,eg\_viscov[1,])  
#> Warning in pseex(model, covdf, ndepths = ndepths, dmax = dmax, addCI = addCI):  
#> Variance and CI calculation when marginalising over `pos` is not yet  
#> implemented.

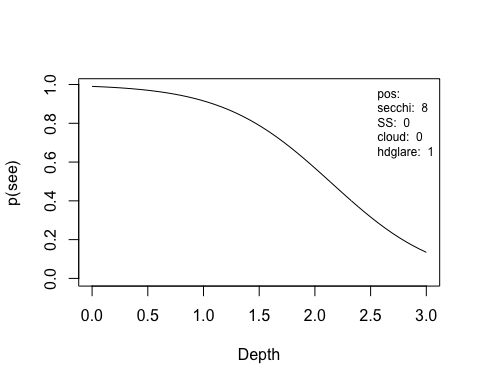


You can also choose not to plot the 95% CI lines by using the argument addCI, and then you don’t get the CI lines or the warning:

plotdetfn(vismodel,eg\_viscov\_withpos[1,],addCI=FALSE)



plotdetfn(vismodel,eg\_viscov[1,],addCI=FALSE)



### Getting the values out of the plot

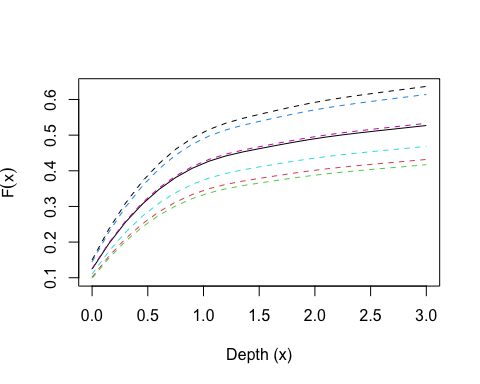
If you want the x- and y- values used in the plots you just assign the call to plotdetfn to some variable. For example:

detfun = plotdetfn(vismodel,eg\_viscov\_withpos[1,],addCI=FALSE,doplot=FALSE)  
detfun[1:10,]  
#> depth p lcl ucl  
#> 1 0.00000000 0.9960500 NA NA  
#> 2 0.06122449 0.9954549 NA NA  
#> 3 0.12244898 0.9947725 NA NA  
#> 4 0.18367347 0.9939906 NA NA  
#> 5 0.24489796 0.9930949 NA NA  
#> 6 0.30612245 0.9920697 NA NA  
#> 7 0.36734694 0.9908971 NA NA  
#> 8 0.42857143 0.9895568 NA NA  
#> 9 0.48979592 0.9880268 NA NA  
#> 10 0.55102041 0.9862827 NA NA

# Plotting the tag/availability model

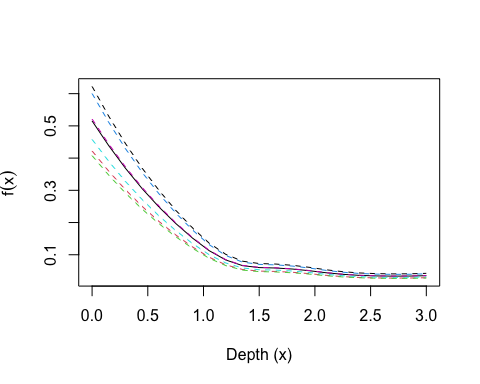
You can also plot the tag (or availability) model - either the cumulative distributon function or the probability distribution function, using the function plotavail. For example, here’s how to create a plot of (solid line) with overlaid with the for each row in eg\_tagcov (dashed lines)

data("eg\_tagcov")  
data("tagmodel")  
depths = seq(0,3,length=21)  
n = nrow(eg\_tagcov)  
CDF = matrix(rep(NA,n\*length(depths)),nrow=n)  
CDF0 = plotavail(depths,tagmodel=NULL,covdf=NULL,what="CDF",doplot=FALSE)  
for(i in 1:n) {  
 CDF[i,] = plotavail(depths,tagmodel=tagmodel,covdf=eg\_tagcov[i,],what="CDF",doplot=FALSE)$F  
}  
ylim.F = range(CDF0$F0,CDF)  
plot(depths,CDF0$F0,xlab="Depth (x)",ylab=expression(F(x)),ylim=ylim.F,type="l")  
for(i in 1:n) lines(depths,CDF[i,],lty=2,col=i)



And here’s how to create a plot of (solid line) with overlaid with the for each row in eg\_tagcov (dashed lines)

pdf = matrix(rep(NA,n\*length(depths)),nrow=n)  
pdf0 = plotavail(depths,tagmodel=NULL,covdf=NULL,what="PDF",doplot=FALSE)  
for(i in 1:n) {  
 pdf[i,] = plotavail(depths,tagmodel=tagmodel,covdf=eg\_tagcov[i,],what="PDF",doplot=FALSE)$f  
}  
ylim.f = range(pdf0$f0,pdf)  
plot(depths,pdf0$f0,xlab="Depth (x)",ylab=expression(f(x)),ylim=ylim.f,type="l")  
for(i in 1:n) lines(depths,pdf[i,],lty=2,col=i)



If you change doplot=FALSE to doplot=TRUE when calling plotavail you get a plot each time plotavail is called.

### Getting the values out of the plot

When you assign the call to the function plotavail to some variable (e.g., to CDF, or CDF0, or PDF, or PDF0 above) you get a data frame with the coordinates used to make the plot. For example:

CDF0  
#> $depth  
#> [1] 0.00 0.15 0.30 0.45 0.60 0.75 0.90 1.05 1.20 1.35 1.50 1.65 1.80 1.95 2.10  
#> [16] 2.25 2.40 2.55 2.70 2.85 3.00  
#>   
#> $F0  
#> [1] 0.1240000 0.1954332 0.2558391 0.3062437 0.3476643 0.3810185 0.4071594  
#> [8] 0.4269443 0.4415528 0.4526640 0.4620000 0.4709477 0.4795541 0.4875308  
#> [15] 0.4946158 0.5008508 0.5064731 0.5117231 0.5168016 0.5218489 0.5270000

## Help

The command help(package="predictp) or just ?predictp gets you the package’s help files. Click index at the bottom of this help page to see the functions for which help is available.