Polar bear example: random slopes and hierarchical smooths

Natasha Klappstein

2024-01-11

Set-up

First, we will load the packages we need for the analysis. `mgcv` is the main package for fitting the SSFs. `gratia` is full of useful tools for visualising and extracting `mgcv` outputs.

```
library(mgcv)
library(cowplot)
library(ggplot2)
library(gratia)
```

We will be analysing GPS locations from 13 adult polar bears in the Beaufort Sea (provided by Andrew Derocher). The data have already been processed to include random steps, and the `stratum` column indicates the step ID and `obs` indicates whether the location is an observed location (1) or random location (0). `step` are the step lengths (km), `angle` are turning angles (radians), and `ice_conc` is the ice concentration (%).

```
# load location data
data <- readRDS("data/polar_bear.RData")
head(data)
```

```
     ID stratum obs       step      angle  ice_conc times
1 bear1       3   1  0.4157609  0.2253824  98.08898     1
2 bear1       3   0  0.3221453  2.8254054  98.06630     1
3 bear1       3   0  1.0301314  0.3372053  98.37928     1
4 bear1       3   0  1.8755764 -2.0743569  96.67261     1
5 bear1       3   0 12.0021570 -0.2243531 100.00000     1
6 bear1       3   0  6.1102090 -0.6898324  99.56963     1
```

The objective of the analysis is to account for inter-individual variability in selection for ice concentration using two methods: i) a random slope model which assumes linear patterns of selection for all individuals, and ii) a hierarchical smooth which allows individuals to have non-linear patterns of selection. For both, we need to define a variable for times, which can be set to 1 for all rows.

```
# create a dummy variable for times
data$times <- 1
```

Fit random slope model

First, we will fit a random slope model. We are modelling step lengths with a gamma distribution (so we include step and its log as covariates), and turning angles with a von Mises distribution (so we include the cosine of the turning angle). Then, we include a global (linear) term for ice concentration, as well as the random slopes (with `s(ice_conc, ID, bs = "re")`).

```
# fit with random slopes
fit_slopes <- gam(cbind(times, stratum) ~
                    step +
                    log(step) +
```

```
                    cos(angle) +
                    ice_conc +
                    s(ice_conc, ID, bs = "re") ,
                  data = data,
                  family = cox.ph,
                  weights = obs)
```

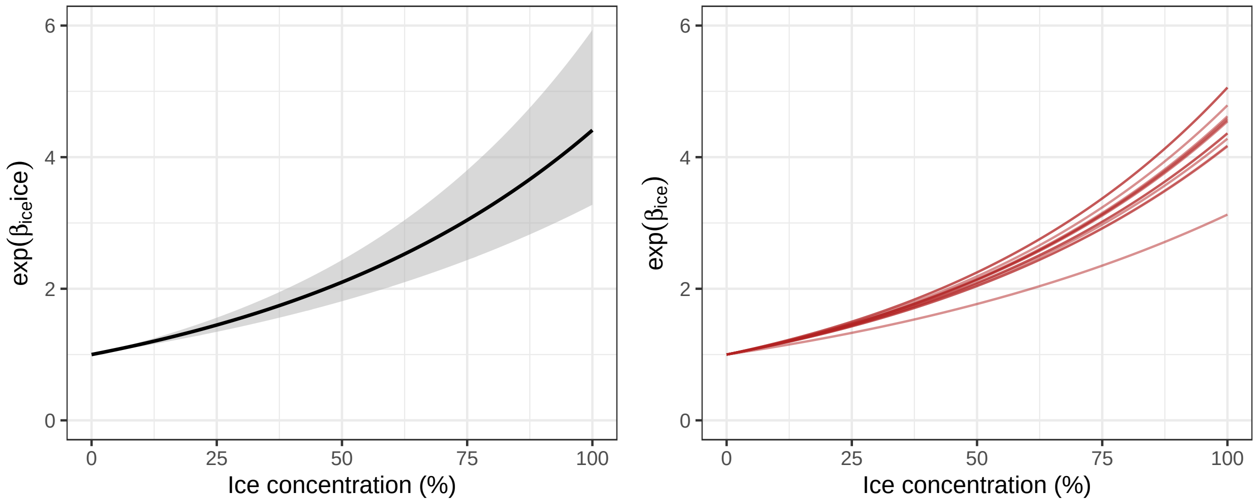
We can look at the summary of the model object.

```
summary(fit_slopes)
```

```

Family: Cox PH
Link function: identity

Formula:
cbind(times, stratum) ~ step + log(step) + cos(angle) + ice_conc +
    s(ice_conc, ID, bs = "re")

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
step        0.117507   0.003371  34.857   <2e-16 ***
log(step)  -0.402598   0.009385 -42.896   <2e-16 ***
cos(angle)  1.043274   0.014303  72.938   <2e-16 ***
ice_conc    0.014839   0.001515   9.793   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                 edf Ref.df Chi.sq p-value
s(ice_conc,ID) 2.653     12  3.981   0.131

Deviance explained =  4.9%
-REML =  40495  Scale est. = 1         n = 297695
```

We may also want to visualise the relationship, as well as the estimated slopes for each individual. The basic plotting functions in `gratia` and `mgcv` can easily plot the population-level slope, but it requires custom code to plot each random slope. Below, I predict for each individual and plot. I've hidden the actual plotting code in `ggplot2`, as it's quite long but you can check out the code file for that if you need.

```
# define grid of ice concentration to predict over
conc_grid <- seq(0, 100, 1)

# get slopes for each individual
bears <- unique(data$ID)
coefID <- as.vector(coef(fit_slopes)[5:(4+length(bears))]) + coef(fit_slopes)[4]

# predict for each individual
r_slope_ID <- NULL
for(i in 1:13) {
  r_slope <- data.frame(bearID = bears[i], conc = conc_grid, RSS = exp(coefID[i] * conc_grid))
  r_slope_ID <- rbind(r_slope_ID, r_slope)
}
```




```r
# hierarchical smooths
fit_smooths <- gam(cbind(times, stratum) ~
                      step +
                      log(step) +
                      cos(angle) +
                      s(ice_conc, k = 5) +
                      s(ice_conc, ID, k = 5, bs = "fs"),
                    data = data,
                    method = "REML",
                    family = cox.ph,
                    weights = obs)
```

```r
summary(fit_smooths)
```

```

Family: Cox PH
Link function: identity

Formula:
cbind(times, stratum) ~ step + log(step) + cos(angle) + s(ice_conc,
    k = 5) + s(ice_conc, ID, k = 5, bs = "fs")

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
step        0.117749   0.003376   34.87   <2e-16 ***
log(step)  -0.402560   0.009389  -42.88   <2e-16 ***
cos(angle)  1.041732   0.014307   72.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                 edf Ref.df Chi.sq p-value
s(ice_conc)    3.458  3.808 101.89  <2e-16 ***
s(ice_conc,ID) 9.424 51.000  15.09  0.0239 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Deviance explained = 1.49%
```

```
-REML =  40469  Scale est. = 1         n = 297695
```

```
###############################
# make plot of random smooths #
###############################
r_smooths <- smooth_estimates(fit_smooths)
pop_smooth <- r_smooths[c(1:100),]

r_smooth_ID <- NULL
for(i in 1:length(bears)) {

  smooth_sub <- subset(r_smooths,
                       smooth == "s(ice_conc,ID)" & ID == bears[i])

  smooth_sub <- data.frame(ID = bears[i],
                           ice_conc = smooth_sub$ice_conc,
                           est = smooth_sub$est + pop_smooth$est)

  r_smooth_ID <- rbind(r_smooth_ID, smooth_sub)
}
```