Appendix B of the article: Modeling the predation sequence of wolves in sheep depredation hotspots.

Ricardo N. Simon, Clément Calenge, Pierre-Yves Quenette, Nicolas Jean & Nolwenn Drouet-Hoguet.

Contents

1 Introduction

This vignette corresponds to the appendix B of the article of Simon et al. (2024). The aim of this paper is to model the predatory sequence of wolves in sheep depredation hotspots in France from 2017 to 2021. A companion package named `wapat` contains the data and functions used for this paper, and is required to reproduce the calculations in this document. The present document is also available as a vignette of this package. To install this package, first install the package `devtools` and use the function `install_github` to install `wapat`:

```
## If devtools is not yet installed, type
install.packages("devtools")

## Install the package badgertub
devtools::install_github("ClementCalenge/wapat", ref="main")
```

Remark: on Windows, it is required to also install the Rtools (https://cran.r-project.org/bin/windows/Rtools/) on your computer to have a working `devtools` package (see https://www.r-project.org/nosvn/pandoc/devtools.html).

Throughout this vignette, we suppose that the reader is familiar with the models and simulations developed in the main paper. We give the R code used to fit the models and assess the power of our model to detect the effect of preventive measures (livestock guarding dogs, LGD, and fence) on the probabilities of approach and attack by the wolf. We also present additional checks of the fitted models (goodness of fit, MCMC convergence).

2 R code used to fit the model

We first describe the R code used to fit the models describing the depredatory behavior of wolves.

2.1 The data

We load the package containing the code and data:

```
library(wapat)
```

And then, we load the dataset:

```
data(hotspotwolf)
str(hotspotwolf)

## 'data.frame': 338 obs. of  14 variables:
##  $ weather             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ habitat             : int  0 0 0 0 1 1 0 1 1 1 ...
##  $ flock.id            : int  342 44 72 47 10 16 90 150 183 183 ...
##  $ shot                : int  0 1 0 1 1 1 1 1 1 1 ...
##  $ dog.presence        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fence.presence      : int  1 0 1 0 1 1 0 1 1 1 ...
##  $ depredation.pressure: int  1 0 1 1 0 0 0 1 1 1 ...
##  $ approach            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ attack              : num  0 0 0 1 0 0 1 1 0 0 ...
##  $ year                : num  2018 2018 2018 2018 2018 ...
##  $ shotwolf            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ shotappr            : num  0 1 0 0 1 1 0 0 1 1 ...
##  $ shotatta            : num  0 0 0 1 0 0 1 1 0 0 ...
##  $ obs.id              : num  127 10 1 49 11 11 17 1 27 27 ...
```

The dataset `hotspotwolf` is a data.frame containing contains the following variables:

- `weather`: a 0/1 vector indicating whether visibility was clear and overcast (1) or not (0) during the intervention, as assessed by observers.
- `habitat`: a 0/1 vector indicating whether habitat was predominantly closed (1) or open (0) during the intervention, as assessed by observers.
- `flock.id`: an integer vector indicating the ID of the flock protected by the intervention.
- `shot`: a 0/1 vector indicating whether a shot was fired or not during the intervention.
- `dog.presence`: a 0/1 vector indicating the presence (1) or absence (0) of livestock guarding dogs protecting the flock.
- `fence.presence`: a 0/1 vector indicating the presence (1) or absence (0) of fences protecting the flock.
- `depredation.pressure`: a 0/1 vector indicating whether the intervention type was simple, i.e. with a single shooter (0), or reinforced, i.e. with multiple shooters (1).
- `approach`: a 0/1 vector indicating whether the wolf approached the flock (1) or not (0) during the intervention, as assessed by observers.
- `attack`: a 0/1 vector indicating whether the wolf attacked the flock (1) or not (0) during the intervention, as assessed by observers.

- `year`: a numeric vector indicating the year of the intervention.
- `shotwolf`: a 0/1 vector indicating whether a shot was fired (1) or not (0) just after detection.
- `shotappr`: a 0/1 vector indicating whether a shot was fired or not just after an approach was diagnosed.
- `shotatta`: a 0/1 vector indicating whether a shot was fired or not just after an attack was diagnosed.
- `obs.id`: a numeric vector containing the ID of the observer.

We also load the package `nimble`, required for the model fit.

```
library(nimble)
```

2.2 Complete model (with interactions between fences and livestock guarding dogs)

2.2.1 Model fit

We program the complete model (effect of all predictive variables + interactions of effects of livestock guarding dogs and fences on the probabilities of approach and attack by the wolf) with the script language used by the package `nimble`. This code is available as the dataset `ModelWithVariables` in the package. We show this code below:

```
data(ModelWithVariables)
ModelWithVariables

## {
##     sigmayear_sw ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_sw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_sa ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_sa ~ T(dt(0, 1, 1), 0, )
##     sigmayear_st ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_st_raw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_ap ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_ap_raw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_at ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_at ~ T(dt(0, 1, 1), 0, )
##     xi_ap ~ dunif(0, 10)
##     xi_st ~ dunif(0, 10)
##     sigmaobs_ap <- sigmaobs_ap_raw * xi_ap
##     sigmaobs_st <- sigmaobs_st_raw * xi_st
##     for (i in 1:nyear) {
##         yeareff_ap[i] ~ dnorm(0, sd = sigmayear_ap)
##         yeareff_at[i] ~ dnorm(0, sd = sigmayear_at)
##         yeareff_sw[i] ~ dnorm(0, sd = sigmayear_sw)
##         yeareff_sa[i] ~ dnorm(0, sd = sigmayear_sa)
##         yeareff_st[i] ~ dnorm(0, sd = sigmayear_st)
##     }
##     for (i in 1:nobs) {
##         obseff_at[i] ~ dnorm(0, sd = sigmaobs_at)
##         obseff_sw[i] ~ dnorm(0, sd = sigmaobs_sw)
##         obseff_sa[i] ~ dnorm(0, sd = sigmaobs_sa)
##         obseff_st_raw[i] ~ dnorm(0, sd = sigmaobs_st_raw)
```

```
##         obseff_st[i] <- obseff_st_raw[i] * xi_st
##         obseff_ap_raw[i] ~ dnorm(0, sd = sigmaobs_ap_raw)
##         obseff_ap[i] <- obseff_ap_raw[i] * xi_ap
##     }
##     for (i in 1:nvar) {
##         a_ap[i] ~ dnorm(0, sd = 2.5)
##         a_at[i] ~ dnorm(0, sd = 2.5)
##     }
##     a_sw ~ dnorm(0, sd = 2.5)
##     a_sa ~ dnorm(0, sd = 2.5)
##     a_st ~ dnorm(0, sd = 2.5)
##     for (i in 1:n) {
##         logit(pshotwolf[i]) <- a_sw + yeareff_sw[year[i]] + obseff_sw[obs.id[i]]
##         logit(pshotappr[i]) <- a_sa + yeareff_sa[year[i]] + obseff_sa[obs.id[i]]
##         logit(pshotatt[i]) <- a_st + yeareff_st[year[i]] + obseff_st[obs.id[i]]
##         logit(papproach[i]) <- inprod(a_ap[1:nvar], X[i, 1:nvar]) +
##             yeareff_ap[year[i]] + obseff_ap[obs.id[i]]
##         logit(pattack[i]) <- inprod(a_at[1:nvar], X[i, 1:nvar]) +
##             yeareff_at[year[i]] + obseff_at[obs.id[i]]
##         shotwolf[i] ~ dbern(pshotwolf[i])
##         approach[i] ~ dbern((1 - shotwolf[i]) * papproach[i])
##         shotappr[i] ~ dbern((1 - shotwolf[i]) * approach[i] *
##             pshotappr[i])
##         attack[i] ~ dbern((1 - shotwolf[i]) * approach[i] * (1 -
##             shotappr[i]) * pattack[i])
##         shotatta[i] ~ dbern((1 - shotwolf[i]) * approach[i] *
##             (1 - shotappr[i]) * attack[i] * pshotatt[i])
##     }
## }
```

We then prepare the dataset for the model fit with `nimble`. Two lists are required: a list with the dataset (response and predictive variables) and a list with the constants used in the model (number of observations, ID of observers, ID of year, etc.):

```
## List with data
dataWf <- list(shotwolf=hotspotwolf$shotwolf,
               approach=hotspotwolf$approach,
               attack=hotspotwolf$attack,
               shotappr=hotspotwolf$shotappr,
               shotatta=hotspotwolf$shotatta,
               X =cbind(1, hotspotwolf$weather, hotspotwolf$habitat,
                        hotspotwolf$depredation.pressure,
                        hotspotwolf$dog.presence, hotspotwolf$fence.presence,
                        hotspotwolf$dog.presence*hotspotwolf$fence.presence))

## List with constant values
constWf <-  list(n=nrow(hotspotwolf),
                 year=hotspotwolf$year-2016,
                 flock.id=as.numeric(factor(hotspotwolf$flock.id)),
                 nyear=max(hotspotwolf$year-2016))
constWf$obs.id <- hotspotwolf$obs.id
constWf$nobs <- max(hotspotwolf$obs.id)
constWf$nflock <- max(constWf$flock.id)
constWf$nvar <- ncol(dataWf$X)
```

We also have defined a dataset named `initValues` in the package to store the starting values for the parameters of the model:

```
data(initValues)
str(initValues)

## List of 11
##  $ a_ap           : Named num [1:7] 2.495 -0.458 0.391 -1.561 0.334 ...
##   ..- attr(*, "names")= chr [1:7] "a_ap[1]" "a_ap[2]" "a_ap[3]" "a_ap[4]" ...
##  $ a_at           : Named num [1:7] 2.185 0.737 0.96 -1.466 0.227 ...
##   ..- attr(*, "names")= chr [1:7] "a_at[1]" "a_at[2]" "a_at[3]" "a_at[4]" ...
##  $ yeareff_ap     : num [1:5] 0 0 0 0 0
##  $ yeareff_at     : num [1:5] 0 0 0 0 0
##  $ obseff_ap      : num [1:141] 0 0 0 0 0 0 0 0 0 0 ...
##  $ obseff_at      : num [1:141] 0 0 0 0 0 0 0 0 0 0 ...
##  $ sigmayear_at   : num 0.2
##  $ sigmayear_ap   : num 0.2
##  $ sigmaobs_at    : num 0.2
##  $ sigmaobs_ap_raw: num 0.2
##  $ xi_ap          : num 1
```

And finally, we use the function `nimbleMCMC()` from the package `nimble` to fit this model, directly passing the components `dataWf` and `constWF` as arguments to this function. We sample 3 chains of 400000 MCMC samples after a burn-in period of 40000 samples. To save disk space, we thin the chain by recording one sample every 400. WARNING: THIS CALCULATION TAKES A VERY LONG TIME!!!! Note that we have included the results of this calculation as a dataset of the package, so that the reader does not need to launch this function to reproduce further calculations:

```
## For reproducibility
set.seed(77)

mcmcComplete <- nimbleMCMC(code = ModelWithVariables, constants = constWf,
                           data = dataWf, thin=400, inits=initValues,
                           nchains = 3, niter = 440000, nburnin=40000,
                           monitors=c("a_at","a_ap","a_sa","a_st","a_sw",
                                      "sigmayear_ap",
                                      "sigmayear_at",
                                      "sigmayear_sw",
                                      "sigmayear_sa",
                                      "sigmayear_st",
                                      "sigmaobs_at",
                                      "sigmaobs_ap",
                                      "sigmaobs_sw",
                                      "sigmaobs_sa",
                                      "sigmaobs_st"),
                           samplesAsCodaMCMC=TRUE,
                           setSeed=77)
```

The results of this fit (an object of class `mcmc.list`) are available in the dataset `mcmcComplete` of the package:

```
data(mcmcComplete)
str(mcmcComplete)

## List of 3
##  $ chain1: 'mcmc' num [1:1000, 1:27] 3.15 1.97 2.67 3.16 2.38 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:27] "a_ap[1]" "a_ap[2]" "a_ap[3]" "a_ap[4]" ...
```

```
##   ..- attr(*, "mcpar")= num [1:3] 1 1000 1
##  $ chain2: 'mcmc' num [1:1000, 1:27] 2.88 1.66 2.55 5.7 2.13 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:27] "a_ap[1]" "a_ap[2]" "a_ap[3]" "a_ap[4]" ...
##   ..- attr(*, "mcpar")= num [1:3] 1 1000 1
##  $ chain3: 'mcmc' num [1:1000, 1:27] 1.15 2.5 4.51 4.51 2.96 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:27] "a_ap[1]" "a_ap[2]" "a_ap[3]" "a_ap[4]" ...
##   ..- attr(*, "mcpar")= num [1:3] 1 1000 1
##  - attr(*, "class")= chr "mcmc.list"
```

The reader can check visually the convergence of the chain by plotting these elements (we do not show this plot in this document to save space, but the reader is encouraged to display it):

```
library(coda)
par(ask=TRUE) ## press enter for next set of graphs
plot(mcmcComplete)
```

We can check more formally this convergence for the parameters of the model with the diagnostic of Gelman and Rubin (1992) (here, we show only the maximum value of the upper bounds of the confidence interval taken over all the parameters of the model, again to save space in this document, but here too the reader is encouraged to display the complete results of `gelman.diag()`:

```
max(gelman.diag(mcmcComplete, multivariate=FALSE)$psrf[,"Upper C.I."])

## [1] 1.030774
```

These diagnostics are all lower than 1.1, as recommended by Gelman and Rubin (1992). The chain convergence is satisfying.

2.2.2 Parameter estimates

We then show the parameter estimates (as well as the 90% credible intervals on these parameters) for the two response variables of interest (probabilities of approach and attack):

```
mo <- do.call(rbind, mcmcComplete)
estim <- apply(mo,2, \(x)
paste0(round(mean(x),2), " (",
       round(quantile(x,0.05),2), ", ",
       round(quantile(x,0.95),2),")"))

estim <- c(estim[1:7], "",estim[8:14])
nom <- c("Intercept", "weather", "habitat", "depredation.pressure",
         "dog.presence", "fence.presence", "interaction dog.fence")
dfo <- data.frame(Name=c(paste0(nom,"_approach"), "",
                         paste0(nom,"_attack")),
                  Estimation=estim)
row.names(dfo) <- 1:nrow(dfo)
print(dfo)

##                               Name          Estimation
## 1              Intercept_approach   2.84 (0.89, 4.77)
```

```
## 2                weather_approach -0.39 (-1.22, 0.46)
## 3                habitat_approach  0.36 (-0.54, 1.35)
## 4   depredation.pressure_approach -1.02 (-2.49, 0.49)
## 5            dog.presence_approach  0.51 (-1.14, 2.13)
## 6          fence.presence_approach  0.46 (-1.08, 2.15)
## 7  interaction dog.fence_approach    -0.22 (-2, 1.52)
## 8
## 9                Intercept_attack   2.17 (0.04, 4.45)
## 10                 weather_attack  1.15 (-0.33, 2.89)
## 11                 habitat_attack   0.62 (-0.57, 1.8)
## 12   depredation.pressure_attack -1.41 (-3.23, 0.29)
## 13            dog.presence_attack  0.55 (-1.36, 2.47)
## 14          fence.presence_attack  0.24 (-1.74, 2.11)
## 15   interaction dog.fence_attack  -0.51 (-2.62, 1.6)
```

This table shows that the interaction between the effects of livestock guarding dogs and fences is not significantly different from 0 (the credible interval includes 0). Note that the values for the interactions (and credible intervals) are displayed in the last row of Tab. 3 in the paper.

It is difficult to interpret the simple effects of the preventive measures when interactions are included in a model. For this reason, we fit again this model without the interaction.

2.3 Model without the interaction between fences and livestock guarding dogs

2.3.1 Model fit

We fit again the model, without the interactions. We therefore change the list containing the reponse and predictive variables, defining again the matrix of explanatory variables. We also update the starting values and the number of variables in the list of constant values:

```
## We change the matrix of predictive variable (we remove the
## interactions)
dataWf2 <- dataWf
dataWf2$X <- cbind(1, hotspotwolf$weather, hotspotwolf$habitat,
                   hotspotwolf$depredation.pressure,
                   hotspotwolf$dog.presence,
                   hotspotwolf$fence.presence)

## We change the starting values, removing the starting value for the
## interactions
initsf2 <- initValues
initsf2$a_ap <- initValues$a_ap[-length(initValues$a_ap)]
initsf2$a_at <- initValues$a_at[-length(initValues$a_at)]

## Change the number of variables
constWf$nvar <- ncol(dataWf2$X)
```

We then fit again the model (WARNING: THIS CALCULATION TAKES A VERY LONG TIME!!!!). Note that we also have included the results of this calculation as a dataset of the package, so that the reader does not need to launch this function to reproduce further calculations:

```
## For reproducibility
set.seed(77)
```

```
mcmcAdditive <- nimbleMCMC(code = ModelWithVariables, constants = constWf,
                           data = dataWf2, thin=400, inits=initsf2,
                           nchains = 3, niter = 440000, nburnin=40000,
                           monitors=c("a_at","a_ap","a_sa","a_st","a_sw",
                                      "sigmayear_ap",
                                      "sigmayear_at",
                                      "sigmayear_sw",
                                      "sigmayear_sa",
                                      "sigmayear_st",
                                      "sigmaobs_at",
                                      "sigmaobs_ap",
                                      "sigmaobs_sw",
                                      "sigmaobs_sa",
                                      "sigmaobs_st"),
                           samplesAsCodaMCMC=TRUE, setSeed=77)
```

The results of this fit are available in the dataset `mcmcAdditive` of the package:

```
data(mcmcAdditive)
```

For this model too, the reader can check visually the convergence of the chain by plotting these elements (we do not show this plot in this document to save some space):

```
par(ask=TRUE) ## press enter for next set of graphs
plot(mcmcAdditive)
```

We can check more formally this convergence for the parameters of the model with the diagnostic of Gelman and Rubin (1992) (here, we show only the maximum value of the upper bounds of the confidence interval on this diagnostic, again to save space in this document):

```
max(gelman.diag(mcmcAdditive, multivariate=FALSE)$psrf[,"Upper C.I."])

## [1] 1.032367
```

These diagnostics are all lower than 1.1, as recommended by Gelman and Rubin (1992). The chain convergence is satisfying.

2.3.2 Parameter estimates

We then show the parameter estimates (as well as the 90% credible intervals on these parameters) for the two response variables of interest (probability of approach and attack):

```
mo <- do.call(rbind, mcmcAdditive)
estim <- apply(mo,2, \(x)
paste0(round(mean(x),2), " (",
       round(quantile(x,0.05),2), ", ",
       round(quantile(x,0.95),2),")"))
estim <- c(estim[1:6], "",estim[7:12])
nom <- c("Intercept", "weather", "habitat",
         "depredation.pressure",
         "dog.presence", "fence.presence")
dfo <- data.frame(Name=c(paste0(nom,"_approach"), "",
                         paste0(nom,"_attack")),
```

```
                  Estimation=estim)
row.names(dfo) <- 1:nrow(dfo)
print(dfo)
```

```
##                              Name          Estimation
## 1             Intercept_approach    2.9 (1.18, 4.65)
## 2               weather_approach -0.38 (-1.18, 0.46)
## 3               habitat_approach  0.38 (-0.46, 1.29)
## 4  depredation.pressure_approach -1.04 (-2.42, 0.37)
## 5           dog.presence_approach  0.33 (-0.58, 1.28)
## 6         fence.presence_approach   0.28 (-0.7, 1.26)
## 7
## 8               Intercept_attack   2.33 (0.24, 4.48)
## 9                 weather_attack  1.12 (-0.39, 2.79)
## 10                habitat_attack  0.66 (-0.46, 1.85)
## 11   depredation.pressure_attack -1.37 (-3.03, 0.25)
## 12            dog.presence_attack  0.21 (-1.08, 1.48)
## 13          fence.presence_attack  -0.1 (-1.32, 1.11)
```

Note that this table corresponds to Tab. 3 from the paper (except for the coefficients of the interactions between fences and dogs, which are taken from previous section). All these coefficients are characterized by a credible interval that includes 0.

2.4 Final model

2.4.1 Model fit

We therefore fit this model again without any variable (keeping only the random effects for the year and observers). The Nimble code for this final model is available as a dataset in the package:

```
data(finalModel)
finalModel
```

```
## {
##     sigmayear_sw ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_sw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_sa ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_sa ~ T(dt(0, 1, 1), 0, )
##     sigmayear_st ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_st_raw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_ap ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_ap_raw ~ T(dt(0, 1, 1), 0, )
##     sigmayear_at ~ T(dt(0, 1, 1), 0, )
##     sigmaobs_at ~ T(dt(0, 1, 1), 0, )
##     xi_ap ~ dunif(0, 10)
##     xi_st ~ dunif(0, 10)
##     sigmaobs_ap <- sigmaobs_ap_raw * xi_ap
##     sigmaobs_st <- sigmaobs_st_raw * xi_st
##     for (i in 1:nyear) {
##         yeareff_ap[i] ~ dnorm(0, sd = sigmayear_ap)
##         yeareff_at[i] ~ dnorm(0, sd = sigmayear_at)
##         yeareff_sw[i] ~ dnorm(0, sd = sigmayear_sw)
##         yeareff_sa[i] ~ dnorm(0, sd = sigmayear_sa)
##         yeareff_st[i] ~ dnorm(0, sd = sigmayear_st)
##     }
```

```
##     for (i in 1:nobs) {
##         obseff_at[i] ~ dnorm(0, sd = sigmaobs_at)
##         obseff_sw[i] ~ dnorm(0, sd = sigmaobs_sw)
##         obseff_sa[i] ~ dnorm(0, sd = sigmaobs_sa)
##         obseff_st_raw[i] ~ dnorm(0, sd = sigmaobs_st_raw)
##         obseff_st[i] <- obseff_st_raw[i] * xi_st
##         obseff_ap_raw[i] ~ dnorm(0, sd = sigmaobs_ap_raw)
##         obseff_ap[i] <- obseff_ap_raw[i] * xi_ap
##     }
##     a_ap ~ dnorm(0, sd = 2.5)
##     a_at ~ dnorm(0, sd = 2.5)
##     a_sw ~ dnorm(0, sd = 2.5)
##     a_sa ~ dnorm(0, sd = 2.5)
##     a_st ~ dnorm(0, sd = 2.5)
##     for (i in 1:n) {
##         logit(pshotwolf[i]) <- a_sw + yeareff_sw[year[i]] + obseff_sw[obs.id[i]]
##         logit(pshotappr[i]) <- a_sa + yeareff_sa[year[i]] + obseff_sa[obs.id[i]]
##         logit(pshotatt[i]) <- a_st + yeareff_st[year[i]] + obseff_st[obs.id[i]]
##         logit(papproach[i]) <- a_ap + yeareff_ap[year[i]] + obseff_ap[obs.id[i]]
##         logit(pattack[i]) <- a_at + yeareff_at[year[i]] + obseff_at[obs.id[i]]
##         shotwolf[i] ~ dbern(pshotwolf[i])
##         approach[i] ~ dbern((1 - shotwolf[i]) * papproach[i])
##         shotappr[i] ~ dbern((1 - shotwolf[i]) * approach[i] *
##             pshotappr[i])
##         attack[i] ~ dbern((1 - shotwolf[i]) * approach[i] * (1 -
##             shotappr[i]) * pattack[i])
##         shotatta[i] ~ dbern((1 - shotwolf[i]) * approach[i] *
##             (1 - shotappr[i]) * attack[i] * pshotatt[i])
##     }
## }
```

We prepare the data for the fit:

```
initsLast <- initValues
initsLast$a_ap <- initsLast$a_ap[1]
initsLast$a_at <- initsLast$a_at[1]

constWf$nvar <- NULL

dataWf$X <- NULL
```

We fit this final model (WARNING: THIS CALCULATION TAKES A VERY LONG TIME!!!!). Note that we also have included the results of this calculation as a dataset of the package, so that the reader does not need to launch this function to reproduce further calculations:

```
## For reproducibility
set.seed(77)
mcmcFinal <- nimbleMCMC(code = finalModel, constants = constWf,
                        data = dataWf, thin=400, inits=initsLast,
                        nchains = 3, niter = 440000, nburnin=40000,
                        monitors=c("a_at","a_ap","a_sa","a_st","a_sw",
                                   "sigmayear_ap",
                                   "sigmayear_at",
                                   "sigmayear_sw",
                                   "sigmayear_sa",
                                   "sigmayear_st",
                                   "sigmaobs_at",
```

```
                                    "sigmaobs_ap",
                                    "sigmaobs_sw",
                                    "sigmaobs_sa",
                                    "sigmaobs_st",
                                    "obseff_at","obseff_ap","obseff_sw",
                                    "obseff_sa","obseff_st",
                                    "yeareff_at","yeareff_ap","yeareff_sw",
                                    "yeareff_sa","yeareff_st"),
                          samplesAsCodaMCMC=TRUE, setSeed=77)
```

The results of this fit are available in the dataset `mcmcFinal` of the package:

```
data(mcmcFinal)
```

For this model too, the reader can check visually the convergence of the chain by plotting these elements (we omit this plot, to save space in this document):

```
par(ask=TRUE) ## press enter for next set of graphs
plot(mcmcFinal)
```

We can check more formally this convergence for the parameters of the model with the diagnostic of Gelman and Rubin (1992) (similarly, we only show the maximum value of the upper limit on the diagnostic taken over all parameters, to save space in this document):

```
max(gelman.diag(mcmcFinal, multivariate=FALSE)$psrf)

## [1] 1.050146
```

These diagnostics are all lower than 1.1, as recommended by Gelman and Rubin (1992). The chain convergence is satisfying.

2.4.2 Goodness of fit

We used the approach of Gelman and Meng (1996) to check the goodness of fit of this final model: Each MCMC iteration r generated a sampled value $\theta^{(r)}$ of the vector of parameters of the model (intercept, slopes of the variables, and random effects). For each simulated value $\theta^{(r)}$, we simulated a replication of the dataset (i.e., we simulated, for each intervention with an observed wolf, a variable describing whether a shot occurred just after detection; if no shot occurred, we then simulated the value of the binary variable "approach"; if an approach was diagnosed, we then simulated whether a shot occurred just after this diagnostic, etc.). Each simulation was thus carried out with the fitted model parameterized by the vector simulated by the r-th MCMC iteration.

We programmed a function to perform these simulations, named `simulateWolfDataGOF()` (see the help page of this function). THIS CALCULATION TAKES A VERY LONG TIME !!!! Note that we have included the results of this calculation as a dataset of the package, so that the reader does not need to launch this function to reproduce further calculations:

```
## For reproducibility
set.seed(777)
simulatedDatasetGOF <- t(sapply(1:3000,  function(x) {
    cat(x,"\r")
    colSums(simulateWolfDataGOF(x, mcmcFinal, constWf))
}))
```

We load the results of these simulations:

```
data(simulatedDatasetGOF)
```

We then compared summary statistics calculated on the observed dataset (number of interventions with a shot after detection, number of interventions in which a wolf approached, etc.) to the distribution of the same statistics derived from the simulated datasets:

```
ap <- paste0("[",apply(round(apply(simulatedDatasetGOF,2, quantile,c(0.05,0.95))),2,
                       paste0,collapse=", "),"]")
ob <- sapply(colnames(simulatedDatasetGOF), function(x) sum(dataWf[[x]]))
data.frame(observed=ob, Expected=ap)

##          observed  Expected
## shotwolf       95 [58, 245]
## approach      205 [67, 239]
## shotappr      104 [29, 136]
## attack         82 [17, 104]
## shotatta       70  [13, 90]
```

All these comparisons show a correct goodness of fit of the model.

2.4.3 Interpretation of the model

We can now present the value of the parameters of the model:

```
mo <- do.call(rbind, mcmcFinal)
mob <- mo[,-c(grep("yeareff",colnames(mo)),grep("obseff",colnames(mo)))]

## Warning: the model that we fit estimates the probability to have a
## shot after observation, approach and attack.  But we are interested
## in the probability of NO shot. The sign of the intercept must
## therefore be changed:
mob[,c(3:5)] <- -mob[,c(3:5)]

pres <- function(x) {
    paste0(round(mean(x),2), " (",
                  round(quantile(x,0.05),2), ", ",
           round(quantile(x,0.95),2),")")
}
u <- data.frame(Parametre=c("Intercept approach","Intercept attack",
                        "Intercept no shot | approach",
                        "Intercept no shot | attack",
                        "Intercept no shot | observation",
                        "SD obs approach",
                        "SD obs attack",
                        "SD obs no shot | approach",
                        "SD obs no shot | attack",
                        "SD obs no shot | observation",
                        "SD year approach",
                        "SD year attack",
                        "SD year no shot | approach",
                        "SD year no shot | attack",
                        "SD year no shot | observation"),
                Estimation=apply(mob,2, pres))
reord <- c(5,1,3,2,4)
```

```
u <- u[c(reord, reord+5, reord+10),]
row.names(u) <- 1:nrow(u)
u
```

```
##                          Parametre           Estimation
## 1  Intercept no shot | observation   0.47 (-0.84, 1.73)
## 2               Intercept approach     2.47 (0.9, 4.03)
## 3     Intercept no shot | approach   -0.03 (-0.82, 0.8)
## 4                 Intercept attack     1.59 (0.3, 2.85)
## 5       Intercept no shot | attack -2.21 (-3.59, -1.17)
## 6     SD obs no shot | observation    1.13 (0.67, 1.67)
## 7                  SD obs approach     1.6 (0.58, 2.96)
## 8        SD obs no shot | approach    1.77 (1.08, 2.65)
## 9                    SD obs attack    0.84 (0.13, 1.84)
## 10         SD obs no shot | attack     0.65 (0.03, 1.8)
## 11   SD year no shot | observation    1.72 (0.84, 3.12)
## 12                SD year approach    1.01 (0.14, 2.49)
## 13      SD year no shot | approach    0.69 (0.09, 1.62)
## 14                  SD year attack    1.27 (0.33, 2.84)
## 15        SD year no shot | attack     0.84 (0.06, 2.3)
```

This table corresponds to the first three columns of Tab. 4 of the paper. We can use this model to predict the mean probability of each step in the behavior sequence:

```
## Estimate mean proba
probability_approach <-  pres(1/(1+exp(-mo[,"a_ap"])))
probability_attack <-  pres(1/(1+exp(-mo[,"a_at"])))
probability_noshot_before_approach <-  pres(1-1/(1+exp(-mo[,"a_sw"])))
probability_noshot_after_approach <-  pres(1-1/(1+exp(-mo[,"a_sa"])))
probability_noshot_after_attack <-  pres(1-1/(1+exp(-mo[,"a_st"])))

data.frame(Probability=c("Not shot before approach","Approach",
                         "Not shot after approach","Attack","Not shot after attack"),
           Estimation =c(probability_noshot_before_approach,
                         probability_approach,
                         probability_noshot_after_approach,
                         probability_attack,
                         probability_noshot_after_attack))
```

```
##                Probability        Estimation
## 1 Not shot before approach   0.6 (0.3, 0.85)
## 2                 Approach 0.89 (0.71, 0.98)
## 3  Not shot after approach 0.49 (0.31, 0.69)
## 4                   Attack 0.81 (0.58, 0.95)
## 5    Not shot after attack 0.12 (0.03, 0.24)
```

This corresponds to the fourth column of Tab. 4 of the paper. We also estimate the mean probability of each step in the behavior sequence, accounting for the uncertainty caused by the random effects of observers and years:

```
probability_approach2 <-  pres(1/(1+exp(-mo[,"a_ap"]+
                                        rnorm(nrow(mo), 0, mo[,"sigmayear_ap"])+
                                        rnorm(nrow(mo), 0, mo[,"sigmaobs_ap"]))))
probability_attack2 <-  pres(1/(1+exp(-mo[,"a_at"]+
                                      rnorm(nrow(mo), 0, mo[,"sigmayear_at"])+
                                      rnorm(nrow(mo), 0, mo[,"sigmaobs_at"]))))
```

```
probability_noshot_before_approach2 <-
    pres(1-1/(1+exp(-mo[,"a_sw"]+
                    rnorm(nrow(mo), 0, mo[,"sigmayear_sw"])+
                    rnorm(nrow(mo), 0, mo[,"sigmaobs_sw"]))))
probability_noshot_after_approach2 <-
    pres(1-1/(1+exp(-mo[,"a_sa"]+
                    rnorm(nrow(mo), 0, mo[,"sigmayear_sa"])+
                    rnorm(nrow(mo), 0, mo[,"sigmaobs_sa"]))))
probability_noshot_after_attack2 <-
    pres(1-1/(1+exp(-mo[,"a_st"]+
                    rnorm(nrow(mo), 0, mo[,"sigmayear_st"])+
                    rnorm(nrow(mo), 0, mo[,"sigmaobs_st"]))))

data.frame(Probability=c("No shot before approach","Approach","No shot after approach",
                         "Attack","No shot after attack"),
           Estimation =c(probability_noshot_before_approach2, probability_approach2,
                         probability_noshot_after_approach2,
                         probability_attack2,probability_noshot_after_attack2))

##               Probability        Estimation
## 1 No shot before approach 0.58 (0.03, 0.99)
## 2                Approach     0.81 (0.2, 1)
## 3  No shot after approach 0.49 (0.03, 0.97)
## 4                  Attack 0.74 (0.17, 0.99)
## 5    No shot after attack 0.16 (0.01, 0.51)
```

... which corresponds to the last column of Tab. 4 of the paper.

3 Simulations

In this section, we show the code used for the simulations carried out in the paper to assess the power of our model to detect the effect of preventive measures (livestock guarding dogs, fences) on the probabilities of approach and attack by the wolf.

We simulated an effect of a preventive measure (either livestock guarding dogs or fences) on a probability of the behavioral sequence (either approach or attack): the strength of the effect was one of the pre-chosen odds-ratio values: 1, 1/2, 1/3, 1/5, 1/10, 1/30, 1/50 and 1/100 (we only simulate odds-ratio lower than 1, as we expect the preventive measure to reduce the probability of approach and attack). For each preventive measure, each behavior of the wolf (approach and attack) and each strength effect, we simulate 100 datasets using the function `simulateDataPow()`, and we fit the model to each dataset with the function `fitModelPow()`, both from the package `wapat` (see the help page of these functions). Model fit is carried out with MCMC. Note that in this simulation context, the package `rstanarm` is much simpler to use to fit Bayesian regression models than the package `nimble` that was used previously to fit the whole predatory sequence, so that the function `fitModelPow()` relies on `rstanarm`. Note also that since this function is used in a loop that simulates fake datasets with the function `simulateDataPow()`, a problematic situation may occur where the response variable takes only 0's or 1's. In such cases, the function `stan_glmer()` used internally by `fitModelPow()` return an error. For such cases, the function fits the model with the function `jags.model()` from the package `rjags`.

For each one of the 100 datasets/model fits, we store the MCMC samples for the coefficient of the tested preventive measure, and we calculate the 90% credible interval on this parameter. We then calculate the proportion of the 100 credible intervals on the parameter that do not include 0 ($\approx$ probability to detect an effect of this proportion). WARNING: THIS CALCULATION TAKES A VERY LONG TIME (SEVERAL HOURS)!!!!. Note that we also have included the results of this calculation as a dataset of

the package, so that the reader does not need to launch this function to reproduce further calculations:

```
## The tested effects:
effect <- log(c(1,2,3,5,10,30,50,100))
qprop <- c("approach","attack")
qvarp <- c("dog.presence","fence.presence")


powerAnalysis <- list()

## For each preventive measure
for (iv in 1:length(qvarp)) {
    liqp <- list()

    ## For each response variable (behavior)
    for (iq in 1:length(qprop)) {
        lief <- list()

        ## And for each effect strength:
        for (ie in 1:length(effect)) {

            varl <- list()

            ## We simulate 100 datasets and fit 100 models
            for (r in 1:100) {
                cat("#####################################################################\n")
                cat("#####################################################################\n")
                cat("#####################################################################\n")
                cat("#####################################################################\n")
                cat("#####################################################################\n")
                cat("Rep.",r,",eff=",ie,",qp=",iq,",vp=",iv,"\n")

                ## Simulation data:
                sid <- simulateDataPow(effect[ie], qvarp[iv], qprop[iq],
                                       hotspotwolf, mcmcFinal)
                re <- fitModelPow(qprop[iq], qvarp[iv], sid, hotspotwolf)
                varl[[r]] <- re
            }
            ## Stores results
            lief[[ie]] <- do.call(cbind,varl)
        }
        liqp[[iq]] <- lief
        powerAnalysis[[iv]] <- liqp
    }
}


resultsPA <- lapply(1:length(powerAnalysis), function(iv) {
    do.call(cbind,lapply(1:length(powerAnalysis[[iv]]),
                         function(iq) {
                     sapply(1:length(powerAnalysis[[iv]][[iq]]),
                            function(ie) {
                         vl <- powerAnalysis[[iv]][[iq]][[ie]]
                         interv <- apply(vl,2,quantile,c(0.05,0.95))
                         pr <- mean(interv[1,]<=0&0<=interv[2,])
                         return(pr)
                     })
```

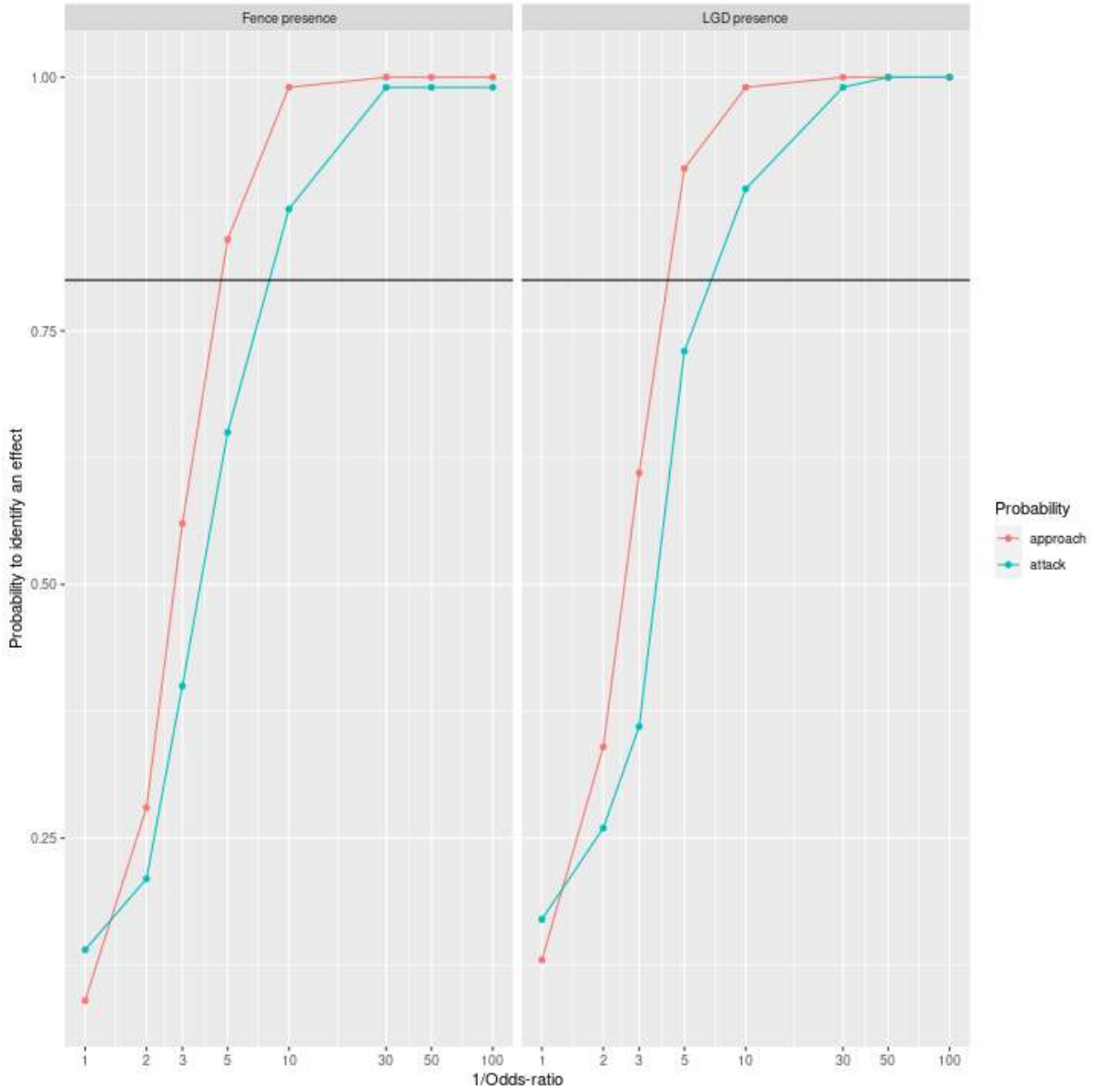
```
                  }))
})
```

The package `wapat` contains a dataset named `resultsPA` with the results of this analysis:

```
data(resultsPA)
str(resultsPA)


## List of 2
##  $ : num [1:8, 1:2] 0.87 0.66 0.39 0.09 0.01 0 0 0 0.83 0.74 ...
##  $ : num [1:8, 1:2] 0.91 0.72 0.44 0.16 0.01 0 0 0 0.86 0.79 ...
```

For each preventive measure, each behavioral response and each effect strength, we show the proportion of the 90% credible intervals on the coefficient that do not include the value 0:

```
library(ggplot2)

## Prepare the data.frame for graphical display with ggplot2:
effet <- c(1,2,3,5,10,30,50,100)
do <- data.frame(var=rep("dog.presence", 2*length(effet)),
                 effet=c(effet,effet),
                 type=c(rep("approach",length(effet)),
                        rep("attack",length(effet))),
                 pr=c(resultsPA[[1]][,1],resultsPA[[1]][,2]))
do2 <- data.frame(var=rep("fence.presence", 2*length(effet)),
                  effet=c(effet,effet),
                  type=c(rep("approach",length(effet)),
                         rep("attack",length(effet))),
                  pr=c(resultsPA[[2]][,1],resultsPA[[2]][,2]))
do <- rbind(do,do2)
do$var <- c(rep("LGD presence",16), rep("Fence presence", 16))
do$Probability <- do$type

## Show the graph
ggplot(do, aes(x=effet,y=1-pr, col=Probability))+geom_point()+
    geom_line()+facet_wrap(~var)+
    geom_hline(yintercept=0.8)+xlab("1/Odds-ratio")+
    ylab("Probability to identify an effect")+
    scale_x_log10(breaks=c(1,2,3,5,10,30,50, 100))
```



This plot corresponds to Fig. 2 of the paper. We can also calculate the inverse odds-ratio of the effect for which we have a probability of 0.8 to detect an effect by interpolating the curves:

```
re <- lapply(c("LGD presence","Fence presence"), function(var) {
    lap <- lapply(c("approach","attack"), function(type) {
        dt <- do[do$var==var&do$type==type,c("effet","pr")]
        k <- 0
        for (i in 1:(nrow(dt)-1)) {
            if (dt$pr[i]>0.2&dt$pr[i+1]<0.2)
                k <- i
        }
        du <- dt[k:(k+1),]
        return(round(approx(du$pr, du$effet, xout=0.2)$y,1))
    })
    names(lap) <- c("approach","attack")
    return(lap)
})
names(re) <- c("LGD presence","Fence presence")
re
```

```
## $`LGD presence`
## $`LGD presence`$approach
## [1] 4.3
##
## $`LGD presence`$attack
## [1] 7.2
##
##
## $`Fence presence`
## $`Fence presence`$approach
## [1] 4.7
##
## $`Fence presence`$attack
## [1] 8.4
```

These values correspond to the values given in the paper.

References

Gelman, A. and Meng, X. 1996. Model checking and model improvement. – In: Gilks, W. and Richardson, S. (eds.), Markov chain Monte Carlo in practice, chap. 11. Chapman & Hall/CRC, pp. 189–201.

Gelman, A. and Rubin, D. 1992. Inference from iterative simulation using multiple sequences. – Statistical Science 7: 457–472.

Simon, R. et al. 2024. Wolves on the hunt in sheep depredation hotspots in france. – in prep.