Example of fit of a model predicting the collisions between the red deer and vehicles in nine French departements

Clement Calenge,
Office national de la classe et de la faune sauvage
Saint Benoist – 78610 Auffargis – France.

October 2016

Contents

1 Introduction

In this vignette, we illustrate the code used by Saint Andrieux et al. (in prep.) to fit the models predicting the number of collisions between three ungulates species and vehicles in nine French departements. We take the example of the red deer here, but the code is exactly the same for the other species. Note that, to preserve copyright on the data, we added a small amount of noise to the data (though we took care to preserve the main structures in the data).

First we load the package `ungulateCollisions` (the dataset `dataCollision` containing the complete data is automatically loaded with the package):

```
library(ungulateCollisions)
```

The structure of the dataset can be displayed with:

```
str(dataCollision)

## List of 3
##  $ RedDeer :List of 5
##   ..$ X          :'data.frame': 101 obs. of  15 variables:
##   .. ..$ forFrag     : num [1:101] 0.232 0.193 -0.175 -0.354 -0.71 ...
##   .. ..$ urbFrag     : num [1:101] 0.0702 -0.18 0.5976 0.0999 -0.7674 ...
##   .. ..$ locr        : num [1:101] 1 2 1 1 1 1 1 1 2 3 ...
##   .. ..$ regr        : num [1:101] 1 3 3 3 2 1 2 3 2 1 ...
##   .. ..$ natr        : num [1:101] 1 3 2 1 1 2 1 1 1 1 ...
##   .. ..$ motr        : num [1:101] 1 3 1 1 1 2 2 1 1 1 ...
##   .. ..$ elev600     : num [1:101] 97 100 97 100 99 99 99 100 100 96 ...
##   .. ..$ elev600_1500: num [1:101] 2 0 2 0 0 0 1 0 0 4 ...
##   .. ..$ elev1500    : num [1:101] 1 0 1 0 1 1 0 0 0 0 ...
##   .. ..$ sinus       : num [1:101] 1 1 1 1 1 2 2 2 1 2 ...
##   .. ..$ hunt        : num [1:101] 12.13 12.99 6.2 7.82 35.15 ...
##   .. ..$ Agriculture : num [1:101] 0.227 0.508 0.282 0.697 0.433 ...
##   .. ..$ Open        : num [1:101] 0.1685 0.137 0.0295 0.0422 0.0607 ...
##   .. ..$ Urban       : num [1:101] 0.0162 0.0911 0.0192 0.0213 0.019 ...
##   .. ..$ Forest      : num [1:101] 0.578 0.264 0.605 0.206 0.487 ...
##   ..$ coll       : Named num [1:101] 1 1 0 0 2 1 0 0 0 0 ...
##   .. ..- attr(*, "names")= chr [1:101] "181" "1810" "1811" "1812" ...
##   ..$ Y          : num [1:101] 2 3 2 2 2 1 2 2 2 2 ...
##   ..$ Area       : num [1:101] 160 211 45 210 61 ...
##   ..$ departement: chr [1:101] "18" "18" "18" "18" ...
##  $ RoeDeer :List of 5
##   ..$ X          :'data.frame': 263 obs. of  15 variables:
##   .. ..$ forFrag     : num [1:263] 0.143 -0.152 -0.272 -0.671 -0.817 ...
##   .. ..$ urbFrag     : num [1:263] -0.0162 -0.0904 0.2846 0.2166 -0.7253 ...
##   .. ..$ locr        : num [1:263] 1 2 1 1 1 1 1 1 2 3 ...
##   .. ..$ regr        : num [1:263] 1 3 3 3 2 1 2 3 2 1 ...
##   .. ..$ natr        : num [1:263] 1 3 2 1 1 2 1 1 1 1 ...
##   .. ..$ motr        : num [1:263] 1 3 1 1 1 2 2 1 1 1 ...
##   .. ..$ elev600     : num [1:263] 100 98 100 99 100 99 100 99 100 100 ...
##   .. ..$ elev600_1500: num [1:263] 0 1 0 0 0 1 0 0 0 0 ...
##   .. ..$ elev1500    : num [1:263] 0 1 0 1 0 0 0 1 0 0 ...
##   .. ..$ sinus       : num [1:263] 2 1 1 1 1 2 2 2 1 2 ...
##   .. ..$ hunt        : num [1:263] 312.8 87.8 86 132.6 152.6 ...
##   .. ..$ Agriculture : num [1:263] 0.251 0.52 0.311 0.712 0.425 ...
##   .. ..$ Open        : num [1:263] 0.1566 0.1493 0.025 0.0342 0.0464 ...
##   .. ..$ Urban       : num [1:263] 0.01227 0.08056 0.05029 0.01596 0.00819 ...
##   .. ..$ Forest      : num [1:263] 0.554 0.222 0.614 0.237 0.52 ...
##   ..$ coll       : Named num [1:263] 11 8 5 9 4 46 20 30 4 16 ...
##   .. ..- attr(*, "names")= chr [1:263] "181" "1810" "1811" "1812" ...
##   ..$ Y          : num [1:263] 2 2 2 2 1 1 2 2 2 2 ...
##   ..$ Area       : num [1:263] 160 210.9 45.2 209.4 61.3 ...
##   ..$ departement: chr [1:263] "18" "18" "18" "18" ...
##  $ WildBoar:List of 5
##   ..$ X          :'data.frame': 242 obs. of  15 variables:
##   .. ..$ forFrag     : num [1:242] 0.15 -0.209 -0.322 -0.594 -0.875 ...
##   .. ..$ urbFrag     : num [1:242] 0.0123 -0.0356 0.2848 0.2458 -0.6878 ...
##   .. ..$ locr        : num [1:242] 1 2 1 1 1 1 1 1 2 3 ...
##   .. ..$ regr        : num [1:242] 1 3 3 3 2 1 2 3 2 1 ...
##   .. ..$ natr        : num [1:242] 1 3 2 1 1 2 1 1 1 1 ...
##   .. ..$ motr        : num [1:242] 1 3 1 1 1 2 2 1 1 1 ...
##   .. ..$ elev600     : num [1:242] 95 99 95 100 96 99 100 97 99 99 ...
##   .. ..$ elev600_1500: num [1:242] 4 0 4 0 4 0 0 2 1 1 ...
##   .. ..$ elev1500    : num [1:242] 0 1 1 0 0 1 0 1 0 0 ...
##   .. ..$ sinus       : num [1:242] 2 1 1 1 1 2 2 2 1 2 ...
##   .. ..$ hunt        : num [1:242] 558.7 69.7 33.6 45.9 63.2 ...
##   .. ..$ Agriculture : num [1:242] 0.217 0.515 0.274 0.69 0.396 ...
##   .. ..$ Open        : num [1:242] 0.1707 0.13053 0.00777 0.0356 0.06119 ...
##   .. ..$ Urban       : num [1:242] 0.00945 0.09287 0.0355 0.02595 0.01274 ...
##   .. ..$ Forest      : num [1:242] 0.545 0.219 0.683 0.249 0.53 ...
##   ..$ coll       : Named num [1:242] 12 4 1 1 1 3 6 8 1 3 ...
##   .. ..- attr(*, "names")= chr [1:242] "181" "1810" "1811" "1812" ...
##   ..$ Y          : num [1:242] 1 3 2 2 2 2 2 2 2 2 ...
##   ..$ Area       : num [1:242] 160.3 210.4 45.2 209.8 61.2 ...
##   ..$ departement: chr [1:242] "18" "18" "18" "18" ...
```

This dataset is a list containing three elements, corresponding to the three focus species `RedDeer, RoeDeer, WildBoar`. Each element is itself a list and contains the following elements:

- `X`: a data.frame containing the variables describing the management units, and supposed to have an effect on the collisions between ungulates and vehicles:
  - `forFrag`: forest fragmentation
  - `urbFrag`: urban fragmentation
  - `locr`: density of local roads
  - `regr`: density of regional roads
  - `natr`: density of national roads
  - `motr`: density of motorways
  - `elev600,elev600_1500,elev1500`: percentage of the management unit covered by areas with elevation respectively lower than 600 m asl, comprised between 600 m and 1500 m asl, and greater than 1500 m asl
  - `sinus`: average sinuosity of the roads in the management unit
  - `hunt`: the hunting bag
  - `Agriculture,Open,Urban,Forest`: proportion of the management unit covered by these habitat types.

- `coll`: the number of collisions between individuals of the species and vehicles in the management unit during the study period;

- `Y`: the number of years during which the collisions with the species have been recorded in the management units;
- `Area`: the area of the management units;
- `departement`: the departement to which each management unit belongs.

In the following sections, we will reproduce the modelling approach described in Saint-Andrieux et al. (in prep.)

2 Bayesian variable selection

2.1 Description of the model

We first describe how we implemented the Bayesian variable selection approach. Let us recall the structure of this model here. For a given species, let N_i be the number of collisions with a vehicle in the management unit i. We assumed that this variable could be described by the following over-dispersed Poisson distribution:

$$\begin{aligned} N_i &\sim \mathcal{P}(\epsilon_i \times \lambda_i \times Y_i \times S_i) \\ \log \epsilon_i &\sim \mathcal{N}(0, \sigma) \end{aligned}$$

where Y_i is the number of years of data available in the management unit i, S_i is the area of the management unit i, λ_i is the average number of collisions per unit area and per year expected under our model (see below) and ϵ_i is a normal over-dispersion residual with zero mean and standard deviation equals to σ.

The average number of collisions per unit area and per year in a management unit i was modeled as a function of the P variables contained in the data.frame `X` stored in each component of the dataset `dataCollision` (see previous section), according to the following log-linear model:

$$\begin{aligned} \log \lambda_i &= \beta_0 + \sum_{j=1}^{P} \alpha_j \times \beta_j \times X_{ij} + \eta_{d(i)} \\ \eta_{d(i)} &\sim \mathcal{N}(0, \sigma_d) \end{aligned}$$

where X_{ij} is the value of the jth variable describing the management unit i, η_d is a random effect describing the effect of the department d, $d(i)$ is the department corresponding to the management unit i, and α_j and β_j are two coefficients characterizing the role of the jth variable in this linear combination: (i) the coefficient α_j can only take values 0 and 1. When this coefficient is equal to 1, the jth variable belongs to the model; when this coefficient is equal to 0, the jth variable does not belong to the model. In a Bayesian context, the value of this coefficient is therefore considered as the realization of a Bernoulli variable characterized by a probability p_j, which is the probability that this variable belongs to the model; (ii) the coefficient β_j can take any real value, and determines the importance of the jth variable on the average number of collisions when this variable belongs to the model, as in a classical regression model. This approach consists in separating the presence of a variable in a model from its importance, and then to estimate the probability of presence of each variable in the model from the data, as suggested by Kuo and Mallik (1998).

We set the following vague priors on the coefficients of the model:

$$\begin{aligned} \beta_0 &\sim \mathcal{N}(0, 100) \\ \beta_j &\sim \mathcal{N}(0, 100) \\ \alpha_j &\sim \mathcal{B}(0.5) \\ \sigma &\sim \mathcal{G}(0.01, 0.01) \\ \sigma_d &\sim \mathcal{G}(0.01, 0.01) \end{aligned}$$

where $\mathcal{B}$ stands for Bernoulli, and $\mathcal{G}$ stands for Gamma.

2.2 Preparation of the data

We then prepare the data for the fit. First, we extract the data.frame containing the explanatory variables in an object named `X`:

```
X <- dataCollision$RedDeer$X
```

Then, note that the three variables describing the elevation are redundant: a given point in a management unit is necessarily either <600 m a.s.l., with an elevation comprised between 600 and 1500 m a.s.l., or $>$ 1500 m a.s.l. Thus, we know that if the elevation of a point is neither >1500 m, nor between 600 m and 1500 m, then it is necessarily $<$ 600 m. We therefore set the variable `elev600` to NULL, to avoid this redundancy:

```
X$elev600 <- NULL
```

We will use the function `prepareFit` from the package `ungulateCollisions` to fit the model. Let us look at the arguments of this function:

```
args(prepareFit)

## function (X, alphas, collisions, nYear, Area, departement)
## NULL
```

The argument names are pretty clear (otherwise, see the help page of this function). Only the argument `alphas` requires an explanation. This argument is a character string vector with length equal to `ncol(X)`. It gives the name, for each variable in the model (included in the data.frame `X` created above), of the group of variables to which each column of `X` belongs. This vector is used to describe how to group the variables so that they are either in the model or out of the model together. Thus, for most columns of `X`, `alphas[i]` will be set equal to `names(X)[i]`, except for the two elevation variables (for which `alphas[i]` will be set to `"elev"`) and the four habitat variables (for which `alphas[i]` will be set to `"hab"`). We create this vector below:

```
## Names of variables
names(X)

##  [1] "forFrag"      "urbFrag"      "locr"
##  [4] "regr"         "natr"         "motr"
##  [7] "elev600_1500" "elev1500"     "sinus"
## [10] "hunt"         "Agriculture"  "Open"
## [13] "Urban"        "Forest"

## Creation of alphas: equal to names(X)
## for most variables...
alphas <- names(X)

## ... except for elevation and habitat,
## which correspond to several variables in X
alphas[7:8] <- "elev"
alphas[11:14] <- "hab"
```

Note that the habitat variables may be strongly correlated together:

```
cor(X[,11:14])

##             Agriculture       Open      Urban     Forest
## Agriculture   1.0000000 -0.1770885  0.0454837 -0.8894876
## Open         -0.1770885  1.0000000 -0.2596748 -0.2305774
## Urban         0.0454837 -0.2596748  1.0000000 -0.1398664
## Forest       -0.8894876 -0.2305774 -0.1398664  1.0000000
```

In particular, the agricultural cover is inversely correlated with the forest habitat (the more agricultural habitat there is in a management unit, and the less forested habitat there is). The presence of correlation between explanatory variables can pose a problem to the Gibbs sampler used to fit the model (e.g. Gilks and Robert 1996). To avoid these correlations, we first rotate the space defined by these four habitat variables using a principal component analysis (which consists in finding new, uncorrelated variables in the space defined by these four habitat variables):

```
## We extract the habitat variables in a table hab
hab <- X[,11:14]

## We delete these columns from the table X
X <- X[,-c(11:14)]

## We perform a PCA using dudi.pca from the package ade4
pc <- ade4::dudi.pca(hab,scannf = FALSE, nf=4)

## The row coordinates are uncorrelated
## this data.frame will be included in the table X
li1 <- pc$l1

## But we first have another transformation to apply:
```

Before storing these new, uncorrelated variable describing the habitat in the table `X`, we first standardize the rest of the table `X`, i.e. we transform it so that each column is characterized by a mean equal to zero and a standard deviation equal to 1. This is a required step to improve the mixing of the Gibbs sampler (Gilks and Roberts 1996):

```
X <- as.data.frame(scale(X))
```

And we finally bind the uncorrelated principal components to this table:

```
X$AX1 <- li1[,1]
X$AX2 <- li1[,2]
X$AX3 <- li1[,3]
X$AX4 <- li1[,4]
```

2.3 Model fit

Finally we structure our data for the fit of the model with the JAGS software. We will use the package `rjags` to fit this model. As explained in the previous section, we use the function `prepareFit`:

```
pf <- prepareFit(X, alphas, dataCollision$RedDeer$coll,
                 dataCollision$RedDeer$Y, dataCollision$RedDeer$Area,
                 dataCollision$RedDeer$departement)

str(pf,1,nchar.max=20)
```

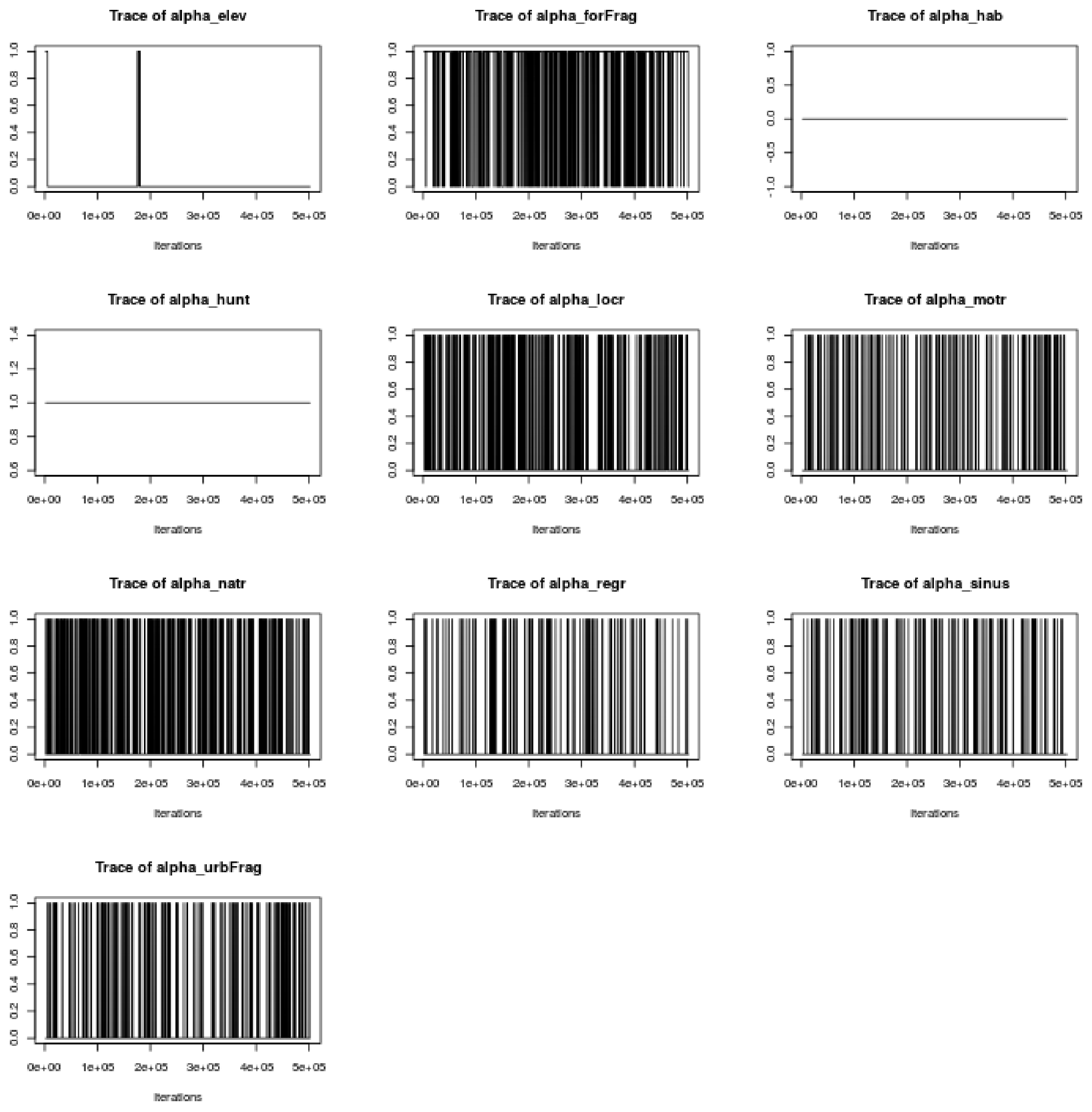
```
## List of 4
##  $ data4jags  :List of 20
##  $ ini        :List of 24
##  $ modelstring: chr "mod"| __truncated__
##  $ coefnames  : chr [1:27] "alpha_forFrag" "alpha_urbFrag" "alpha_locr" "alpha_regr" ...
```

The list `pf` contains all the elements required for the fit. Thus, the code for the JAGS model is stored in the element `modelstring` of this list (use `cat(pf$modelstring)` if you want to see this code), the starting values are stored in the element `ini`, and the data required for the fit are stored in the list `data4jags`. The name of the coefficients to be monitored is stored in the component `coefnames`. It is then possible to fit the model with JAGS using the following code. WARNING: the execution of this code is very long (about one hour)!! We have stored the results of this computation in the dataset `modelRedDeer` to allow the reader to avoid this execution (i.e. no need to execute the following code, the dataset `modelRedDeer` is automatically loaded with the package `ungulateCollisions`):

```
mo <- jags.model(textConnection(pf$modelstring), ini=pf$ini, data=pf$data4jags)
update(mo, n.iter=1000)
modelRedDeer <- coda.samples(mo, variable.names = pf$coefnames,
                             n.iter = 500000, thin=100)
```

The reader can have an idea of the quality of the mixing properties for the parameters alpha by typing:

```
par(mfrow=c(4,3))
traceplot(modelRedDeer[[1]][,grep("alpha", colnames(modelRedDeer[[1]]))])
```



The mixing properties were satisfying (i.e. the chain switches frequently from 0 to 1, when the probability p_j is neither 0 nor 1).

2.4 Results

We can use the function `probabilityKM` to calculate the probability that each variable is included in the true model, as well as the probability that each model (i.e. each combination of variables) is the true model:

```
probabilityKM(modelRedDeer)

## $variables
##    variable  proba
## 1      hunt 1.0000
## 2   forFrag 0.6824
## 3      natr 0.1290
## 4      locr 0.1158
## 5   urbFrag 0.0552
## 6      motr 0.0464
## 7     sinus 0.0290
## 8      regr 0.0236
## 9      elev 0.0146
## 10      hab 0.0000
##
## $models
##                                  model  proba
## 1                         forFrag+hunt 0.4302
## 2                                 hunt 0.2218
## 3                    forFrag+hunt+locr 0.0826
## 4                            hunt+natr 0.0490
## 5                    forFrag+hunt+natr 0.0396
## 6                 forFrag+hunt+urbFrag 0.0380
## 7                    forFrag+hunt+motr 0.0164
## 8                    forFrag+hunt+regr 0.0122
## 9                   forFrag+hunt+sinus 0.0114
## 10              forFrag+hunt+locr+natr 0.0106
## 11                   elev+forFrag+hunt 0.0098
## 12                          hunt+sinus 0.0090
## 13                      hunt+motr+natr 0.0088
## 14              forFrag+hunt+motr+natr 0.0078
## 15                           hunt+locr 0.0056
## 16                           hunt+motr 0.0056
## 17                           hunt+regr 0.0050
## 18                        hunt+urbFrag 0.0050
## 19             forFrag+hunt+locr+sinus 0.0032
## 20           forFrag+hunt+locr+urbFrag 0.0030
## 21              forFrag+hunt+locr+motr 0.0020
## 22              elev+forFrag+hunt+locr 0.0016
## 23           forFrag+hunt+natr+urbFrag 0.0016
## 24                      hunt+locr+natr 0.0016
## 25         forFrag+hunt+locr+motr+natr 0.0014
## 26           forFrag+hunt+regr+urbFrag 0.0012
## 27          forFrag+hunt+sinus+urbFrag 0.0012
## 28           forFrag+hunt+motr+urbFrag 0.0010
## 29                      hunt+natr+regr 0.0010
## 30                   hunt+natr+urbFrag 0.0010
## 31              forFrag+hunt+locr+regr 0.0008
## 32              elev+forFrag+hunt+natr 0.0006
## 33           elev+forFrag+hunt+urbFrag 0.0006
## 34      forFrag+hunt+motr+natr+urbFrag 0.0006
## 35             forFrag+hunt+natr+sinus 0.0006
## 36              elev+forFrag+hunt+regr 0.0004
## 37                      elev+hunt+natr 0.0004
## 38        forFrag+hunt+locr+natr+sinus 0.0004
## 39        forFrag+hunt+motr+natr+sinus 0.0004
## 40              forFrag+hunt+motr+regr 0.0004
## 41              forFrag+hunt+natr+regr 0.0004
## 42             forFrag+hunt+regr+sinus 0.0004
## 43                 hunt+locr+motr+natr 0.0004
## 44              hunt+locr+natr+urbFrag 0.0004
## 45                     hunt+natr+sinus 0.0004
## 46         elev+forFrag+hunt+locr+natr 0.0002
## 47         elev+forFrag+hunt+locr+regr 0.0002
## 48             elev+forFrag+hunt+sinus 0.0002
## 49                           elev+hunt 0.0002
## 50               elev+hunt+natr+urbFrag 0.0002
## 51                    elev+hunt+urbFrag 0.0002
## 52 forFrag+hunt+locr+motr+natr+urbFrag 0.0002
## 53        forFrag+hunt+locr+motr+sinus 0.0002
## 54         forFrag+hunt+locr+natr+regr 0.0002
## 55        forFrag+hunt+locr+regr+sinus 0.0002
## 56     forFrag+hunt+locr+sinus+urbFrag 0.0002
## 57         forFrag+hunt+motr+natr+regr 0.0002
## 58        forFrag+hunt+natr+regr+sinus 0.0002
## 59               hunt+locr+motr+urbFrag 0.0002
## 60                 hunt+locr+natr+sinus 0.0002
## 61                       hunt+locr+regr 0.0002
## 62                      hunt+locr+sinus 0.0002
## 63                 hunt+motr+natr+sinus 0.0002
## 64               hunt+motr+natr+urbFrag 0.0002
## 65                       hunt+motr+regr 0.0002
## 66                 hunt+motr+regr+sinus 0.0002
## 67              hunt+natr+sinus+urbFrag 0.0002
## 68                    hunt+regr+urbFrag 0.0002
```

The results here are approximately identical to those presented by Saint-Andrieux et al. They are not exactly the same, as we added some noise to the data.

Thus, the best model has about one chance out of two to include the forest fragmentation and the hunting bag (as noted by Saint-Andrieux et al.).

3 Final model

3.1 Model fit

We therefore fitted the final model to the data, i.e. the following Bayesian model:

$$\begin{aligned} \log \lambda_i &= \beta_0 + \sum_{j \in B} \beta_j \times X_{ij} + \eta_{d(i)} \\ \eta_{d(i)} &\sim \mathcal{N}(0, \sigma_d) \end{aligned} \tag{1}$$

Where B is the set of variables identified by the Kuo and Mallik's approach (i.e. including only `hunt` and `forFrag`). We use the function `finalModel` (see the help page of this function for more information about its arguments) to prepare the data for this fit:

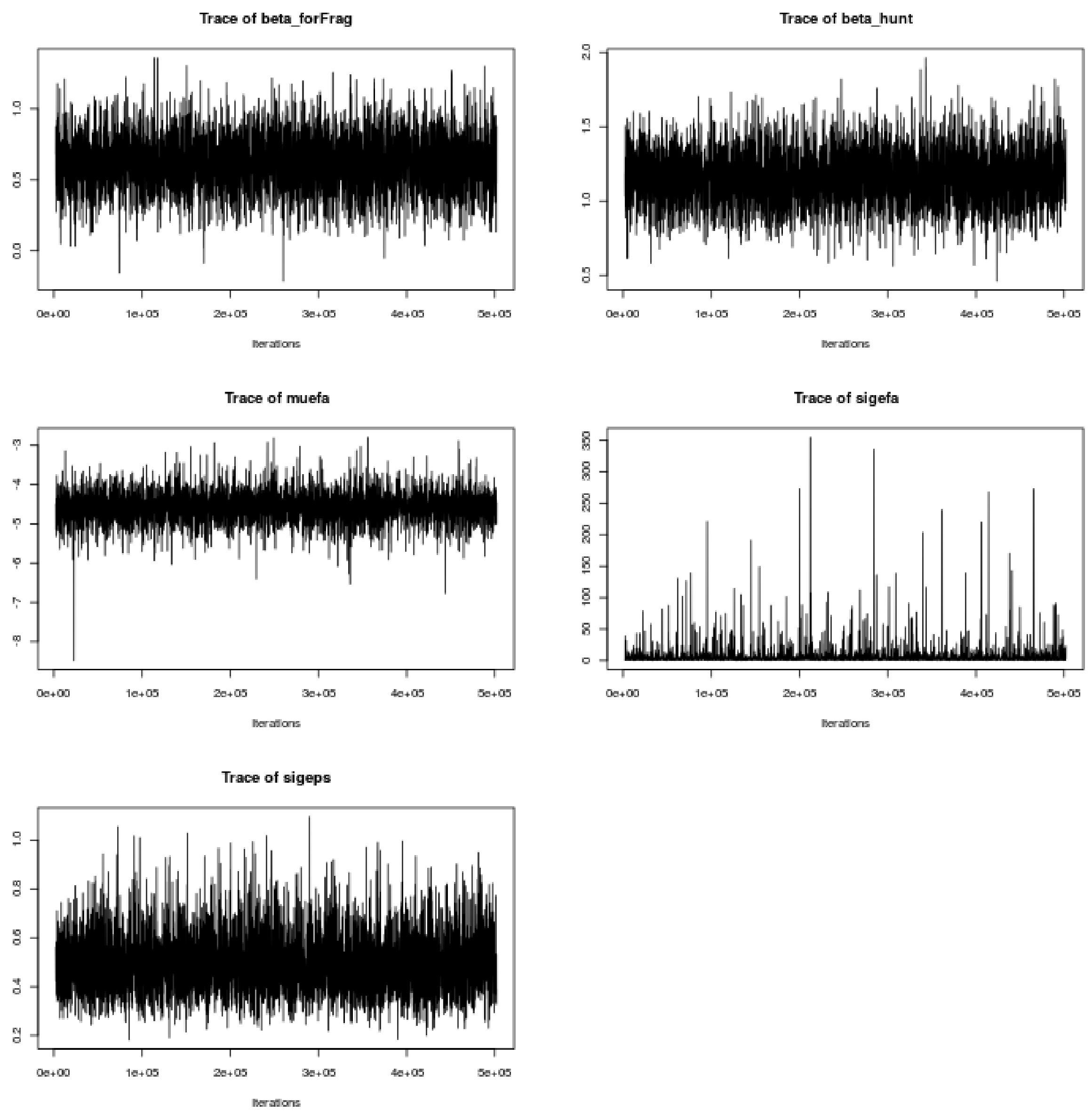
```r
pfm <- finalModel(X, c("forFrag","hunt"), dataCollision$RedDeer$coll,
                  dataCollision$RedDeer$Y, dataCollision$RedDeer$Area,
                  dataCollision$RedDeer$departement)
```

Finally, we can fit the model with JAGS. Again, the execution of this code can be long, so we stored the results in the dataset `finalModelRedDeer`, so that the reader does not necessarily need to execute the following code (the dataset `finalModelRedDeer` is automatically loaded with the package `ungulateCollisions`):

```r
mo <- jags.model(textConnection(pfm$modelstring), ini=pfm$ini, data=pfm$data4jags)
update(mo, n.iter=1000)
finalModelRedDeer <- coda.samples(mo, variable.names = pfm$coefnames,
                                  n.iter = 500000, thin=100)
```

Again, we can have a look at the mixing properties of this model:

```r
par(mfrow = c(3,2))
traceplot(finalModelRedDeer)
```



These properties were satisfying. Note that the diagnostic of Raftery and Lewis (1992) confirm this observation

```
raftery.diag(finalModelRedDeer)

## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##                  Burn-in  Total  Lower bound  Dependence
##                  (M)      (N)    (Nmin)       factor (I)
##   beta_forFrag 200        374100 3746          99.9
##   beta_hunt    200        374100 3746          99.9
##   muefa        200        386600 3746         103.0
##   sigefa       200        386600 3746         103.0
##   sigeps       200        393000 3746         105.0
```

The required number of iterations N were lower than 500000 (the actual number of iterations) for all parameters. Note however that there is a strong dependency between successive values of the Markov chain.

We now look at the goodness of fit of this model. As indicated by Saint-Andrieux et al., We then examined the fit of the model using the approach recommended by Gelman and Meng (1996). For every iteration of the MCMC, we simulated a hypothetical replication of the dataset using equation 1, i.e. we simulated a number of collisions in each management unit. We then compared the observed number of collisions with the statistical distribution of simulated numbers of collisions. We compute these simulated distribution below:

```
## the MCMC iterations:
cf <- as.data.frame(finalModelRedDeer[[1]])
rp <- list()
## For each MCMC iteration
for (r in 1:nrow(cf)) {
    ## progress bar
    cat(round(100*r/nrow(cf)), "\r")
    departement <- as.numeric(factor(dataCollision$RedDeer$departement))
    ## get the parameters relative to departement random effects
    sigefa <- cf$sigefa[r]
    muefa <- cf$muefa[r]
    efal <- rnorm(7, muefa, 1/sqrt(sigefa))
    efal <- efal[departement]
    ## overdispersion residuals
    sigeps <- cf$sigeps[r]
    resid <- rnorm(length(departement),0,1/sqrt(sigeps))
    ## And formula
    ev <- dataCollision$RedDeer$Y*
        dataCollision$RedDeer$Area*
            exp(efal+resid+
                    cf$beta_forFrag[r]*X$forFrag+
                        cf$beta_hunt[r]*X$hunt)

    ## Simulation of the number of collisions
    rp[[r]] <- rpois(length(ev), ev)
}
SimuGOF <- do.call("cbind",rp)
```

We calculated the proportion of the 90% credible intervals that contained the observed number of collisions:

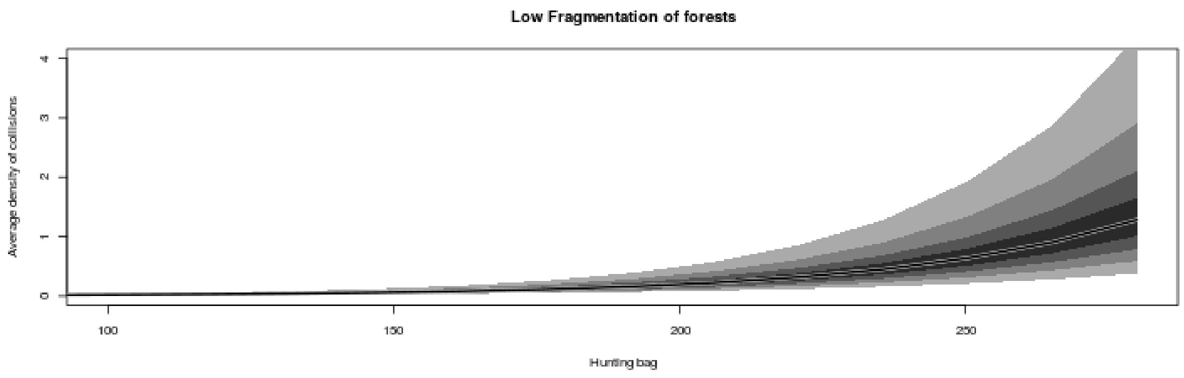
```
aa <- apply(SimuGOF, 1, quantile, c(0.05,0.95))
sum(dataCollision$RedDeer$coll>=aa[1,]&
        dataCollision$RedDeer$coll<=aa[2,])/
    length(dataCollision$RedDeer$coll)

## [1] 0.970297
```

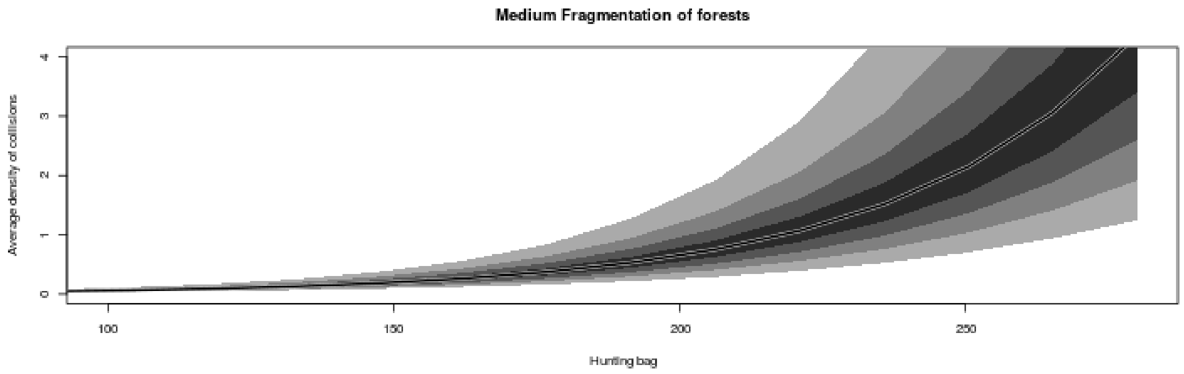
This indicates that the fit of our model was correct for the the red deer. We could therefore validate this model.

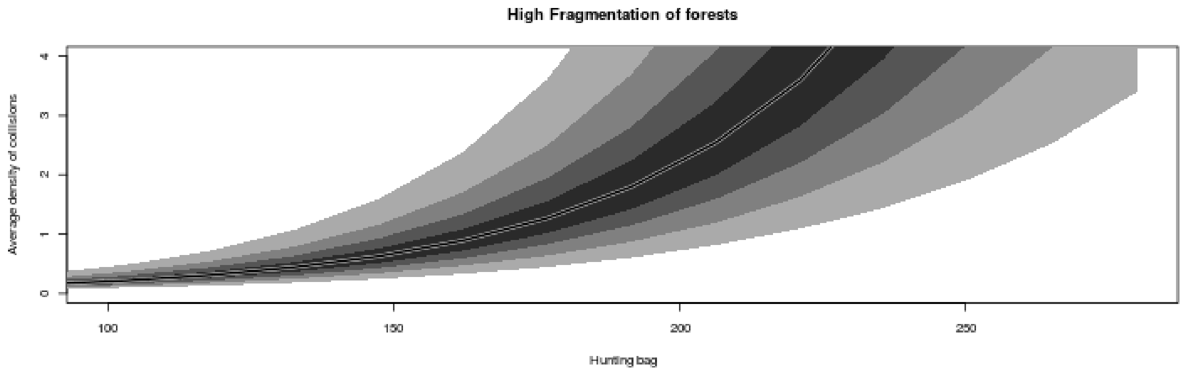
3.2 Graphical display

Finally, we represented graphically the relationship between the number of collisions, the hunting bag and the forest fragmentation:

```r
cf <- as.data.frame(finalModelRedDeer[[1]])

## Explanatory variables for this model
re <- seq(0,280, length=20)
reo <- (re-mean(dataCollision$RedDeer$X$hunt))/
    sd(dataCollision$RedDeer$X$hunt)
ra <- c(c(-1, 0, 1)-mean(dataCollision$RedDeer$X$forFrag))/
    sd(dataCollision$RedDeer$X$forFrag)

## For each MCMC iteration, calculation of the average density of collision
lir <- list()
for(r in 1:nrow(cf)) {
    vr1 <- exp(cf$muefa[r]+cf$beta_hunt[r]*reo + cf$beta_forFrag[r]*ra[1])
    vr2 <- exp(cf$muefa[r]+cf$beta_hunt[r]*reo + cf$beta_forFrag[r]*ra[2])
    vr3 <- exp(cf$muefa[r]+cf$beta_hunt[r]*reo + cf$beta_forFrag[r]*ra[3])
    lir[[r]] <- data.frame(vr1, vr2, vr3)
}

## We represent the credible intervals at various levels
## (10%, 20%, ..., 80%, 90%)
par(mfrow=c(3,1),bg="white")
for (i in 1:3) {
    df <- do.call("cbind",lapply(lir, function(x) x[,i]))
    qu <- (apply(df, 1, quantile, seq(0,1,by=0.1)))
    plot(re, qu[1,], ty="n", ylim = c(0.0001,4),
         xlim=c(100,280), ylab = "Average density of collisions",
         main=paste(c("Low","Medium","High")[i],
         "Fragmentation of forests"),
         xlab="Hunting bag")
    for(j in 2:5) {
        polygon(c(re,rev(re)),
                c(qu[j,], rev(qu[nrow(qu)-j+1,])), col=
                    grey((6-j)/6), border=NA)
    }
    lines(re, qu[6,], lwd=3, col="white")
    lines(re, qu[6,], lwd=2, col="black")
    if (i==1)
        legend(0, 15,
               paste(100-2*seq(10,40, by=10),
                     "%", sep=""),
               fill=grey((6-(2:5))/6))
}
```







The four shades of grey indicate (from darker to lighter shades): 20%, 40%, 60% and 80% credible intervals.

Models for other species can be fitted in a similar way.

4 References

- Gelman, A. and Meng, X. (1996) Model checking and model improvement. In Gilks, W. and Richardson, S. (Eds.) 1996. Markov chain Monte Carlo in practice. Chapman and Hall/CRC, 189-201.

- Gilks, W. and Roberts, G. (1996) Strategies for improving MCMC. In Gilks, W. and Richardson, S. (Eds.) 1996. Markov Chain Monte Carlo in practice. Chapman and Hall/CRC, 89-114.

- Kuo, L. and Mallik, B. (1998) Variable selection for regression models. Sankhya: The Indian Journal of Statistics, Series B, 60, 65-81.

- Raftery, A. and Lewis, S. (1992). Practical markov chain monte carlo: comment: one long run with diagnostics: implementation strategies for Markov Chain Monte Carlo. Statistical Science, 7, 493-497.

- Saint Andrieux, C., Calenge, C. and Bonenfant, C. (in prep). Comparison of ecological, biological and anthropogenic causes of vehicle-wildlife collisions among three large mammalian species.