Adapter les résultats de Harvest models of small populations of a large carnivore using Bayesian forecasting par Andrén et al. 2020 au modèle déjà existant de dynamique des populations de loups en France

Olivier Gimenez & Loïc Pages

12/01/2024

Introduction

L'objectif de ce code est de modéliser la dynamique de population de loup en France et de prédir l'impact de la chasse sur celle-ci. Pour ce faire, on reprend ici le modèle de Andrén et al. 2020. Premièrement, on calcule la viabilité de la population entre les années 1995 à 2021 selon un modèle logistique. Puis on modélise le nombre optimal d'animaux à tuer pour maintenir la viabilité de la population.

Préparatifs

On calcule la viabilité de la population selon un modèle exponentiel qui prend aussi en compte la chasse du loup.

Tout se passe en bayésien. Si vous vous embêtez, vous pouvez m'écouter pendant 7 heures introduire tout ça par ici. Pour ce qui nous intéresse ici, il nous faudra un package spécifique pour implémenter les méthodes MCMC.

```
library(R2jags)
```

```
## Loading required package: rjags

## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs

##
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':
##
##     traceplot
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Les données

```r
harvest <- c(0,0,0,0,0,1,0,0,2,1,2,0,0,1,0,4,4,6,18,36,34,42,51,98,105,103,169)
```

Les estimations d'effectifs par CMR:

```r
CMR <- c(17.1,35.4,
47.7,
25.1,
62.6,
47.9,
81.7,
110.5,
102.7,
135.9,
132.6,
101.7,
130.3,
141.4,
141.5,
175.5,
210.3,
174.5,
353.6,
280.2,
376.7,
561.2,
571.9,
682.4,
645.7,
783.8,
868)
```

On met ensemble les effectifs estimés par CMR ainsi que les nombres de loups tués.

```r
thedata <- cbind(round(CMR), harvest)
colnames(thedata) <- c("N", "H")
thedata <- as.data.frame(thedata)
nyears <- nrow(thedata)
```

Modèle avec les prélèvements

On suit Andrén, H., Hobbs, N. T., Aronsson, M., Brøseth, H., Chapron, G., Linnell, J. D. C., Odden, J., Persson, J., and Nilsen, E. B.. 2020. Harvest models of small populations of a large carnivore using Bayesian forecasting. *Ecological Applications* 30(3):02063. 10.1002/eap.2063.

Dans leur papier, Henrik et les collègues construisent un modèle démographique structuré en classes d'âge. J'ai pas envie de me lancer dans un truc compliqué, l'idée est simplement de comprendre comment dérouler leur approche.

On part sur un modèle exponentiel. On stipule que les effectifs N_t à l'année t sont obtenus à partir des effectifs à l'année $t-1$ auxquels on a retranché les prélèvements H_{t-1}, le tout multiplié par le taux de croissance annuel λ :

$$N_t = \lambda(N_{t-1} - H_{t-1}).$$

Cette relation est déterministe. Pour ajouter de la variabilité démographique, on suppose que les effectifs sont distribués selon une distribution log-normale, autrement dit que les effectifs sont normalement distribués sur l'échelle log :

$$\log(N_t) \sim \text{Normale}(\mu_t, \sigma_{\text{proc}})$$

avec $\mu_t = \log(N_t) = \log(\lambda(N_{t-1} - H_{t-1}))$ et σ_{proc} l'erreur standard des effectifs sur l'échelle log. On aurait pu prendre une loi de Poisson à la place. La stochasticité environnementale est en général captée par le taux de croissance, mais pas ici puisqu'il est constant. C'est une hypothèse forte du modèle. Dans l'idéal, on pourrait coupler le modèle de capture-recapture, et le modèle qui décrit l'évolution des effectifs au cours du temps.

On ajoute une couche d'observation qui capture les erreurs sur les effectifs. Si l'on note y_t les effectifs observés, on suppose que ces comptages annuels sont distribués comme une loi de Poisson de moyenne les vrais effectifs N_t:

$$y_t \sim \text{Poisson}(N_t).$$

```r
model <- function(){

  # Priors

  sigmaProc ~ dunif(0, 10)
  tauProc <- 1/sigmaProc^2
  lambda ~ dunif(0, 5)

  N[1] ~ dgamma(1.0E-6, 1.0E-6)

  # Process model
  for (t in 2:(nyears)) {
    mu[t] <- lambda * (N[t-1] - harvest[t-1])
    Nproc[t] <- log(max(1, mu[t]))
    N[t] ~ dlnorm(Nproc[t], tauProc)
  }

  # Observation model
  for (t in 1:nyears) {
```

```
    y[t] ~ dpois(N[t])
  }

  # Projected population
  for (t in (nyears + 1):(nyears + 2)) {
    Nproc[t] <- log(max(1, lambda*(N[t-1])))
    N[t] ~ dlnorm(Nproc[t], tauProc)
  }
}
```

On prépare les données.

```
bugs.data <- list(
    nyears = nrow(thedata),
    y = round(thedata$N),
    harvest = thedata$H)
```

On précise les paramètres à estimer et le nombre de chaines de MCMC (j'en prends trois ici).

```
bugs.monitor <- c("lambda", "sigmaProc","N", "tauProc")
bugs.chains <- 3
bugs.inits <- function(){
    list(
    )
}
```

Allez zooh, on lance la machine!

```
library(R2jags)
wolf_mod <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model,
                  n.chains = bugs.chains,
                                                    n.thin = 10,
                                                    n.iter = 100000,
                                                    n.burnin = 50000)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 31
##    Total graph size: 204
##
## Initializing model
```

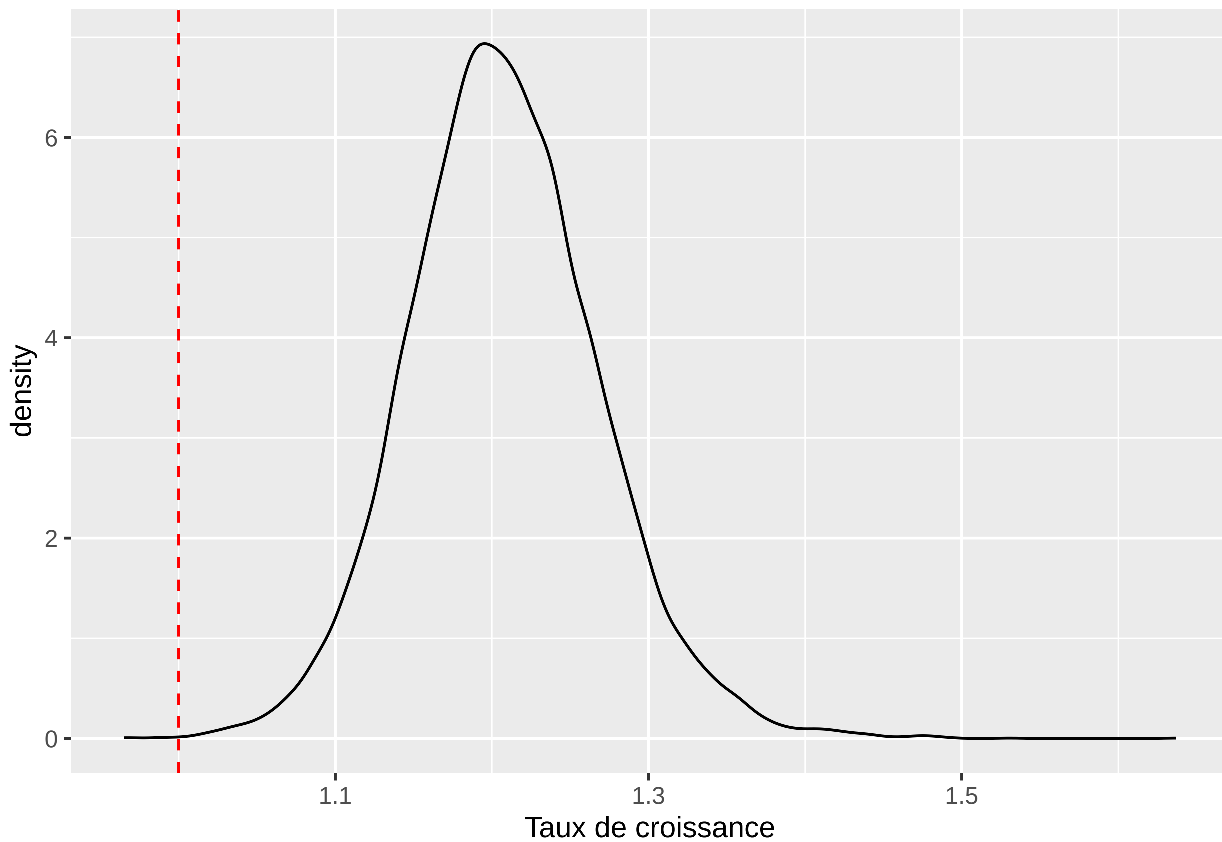
Jetons un coup d'oeil aux estimations.

```r
print(wolf_mod, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/tmp/RtmpurfewI/model167f5be78649.txt", fit using jags,
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##            mu.vect sd.vect    2.5%      50%    97.5%  Rhat n.eff
## N[1]        21.054   3.778  14.375   20.788   29.065 1.001 15000
## N[2]        32.029   4.387  24.228   31.709   41.486 1.001 15000
## N[3]        41.169   5.168  32.010   40.860   52.155 1.001 15000
## N[4]        35.174   4.760  26.247   35.074   44.781 1.001 15000
## N[5]        55.344   6.240  44.014   55.014   68.398 1.001 15000
## N[6]        54.830   6.166  43.340   54.623   67.405 1.001  7100
## N[7]        79.982   7.732  65.714   79.757   95.787 1.001 15000
## N[8]       105.456   9.142  88.508  105.127  124.360 1.001  8400
## N[9]       106.733   9.012  89.653  106.448  124.840 1.001 15000
## N[10]      131.821  10.454 112.573  131.574  153.181 1.001  8400
## N[11]      130.021  10.268 110.683  129.688  151.366 1.001  8000
## N[12]      108.085   9.295  90.729  107.812  127.098 1.001 15000
## N[13]      128.704  10.294 109.604  128.307  149.998 1.001 15000
## N[14]      140.249  10.647 120.007  139.992  161.630 1.001 15000
## N[15]      144.841  10.836 124.627  144.646  166.999 1.001 15000
## N[16]      175.806  12.306 152.732  175.486  201.008 1.001 13000
## N[17]      205.475  13.494 179.640  205.080  232.944 1.001 15000
## N[18]      186.835  13.147 161.760  186.616  213.215 1.001 15000
## N[19]      340.609  18.188 305.691  340.360  377.153 1.001 15000
## N[20]      289.772  16.214 258.547  289.603  322.224 1.001 13000
## N[21]      378.927  18.633 343.525  378.784  416.018 1.001  5800
## N[22]      554.591  22.954 511.342  554.046  600.711 1.001 12000
## N[23]      574.759  23.199 529.911  574.661  620.474 1.001 15000
## N[24]      679.077  25.151 630.405  678.839  729.318 1.001 14000
## N[25]      650.871  24.736 603.219  650.749  700.253 1.001  7300
## N[26]      782.135  27.178 730.035  781.852  836.675 1.001 15000
## N[27]      866.975  29.460 810.022  867.014  925.649 1.001 15000
## N[28]     1077.492 286.082 621.561 1041.181 1740.149 1.001 15000
## N[29]     1350.976 549.688 587.969 1252.334 2673.666 1.001 15000
## lambda       1.208   0.061   1.093    1.205    1.335 1.001 15000
## sigmaProc    0.250   0.054   0.163    0.243    0.375 1.001 15000
## tauProc     18.341   7.884   7.124   16.932   37.475 1.001 15000
## deviance   218.858   8.174 204.607  218.148  236.520 1.001 15000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 33.4 and DIC = 252.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

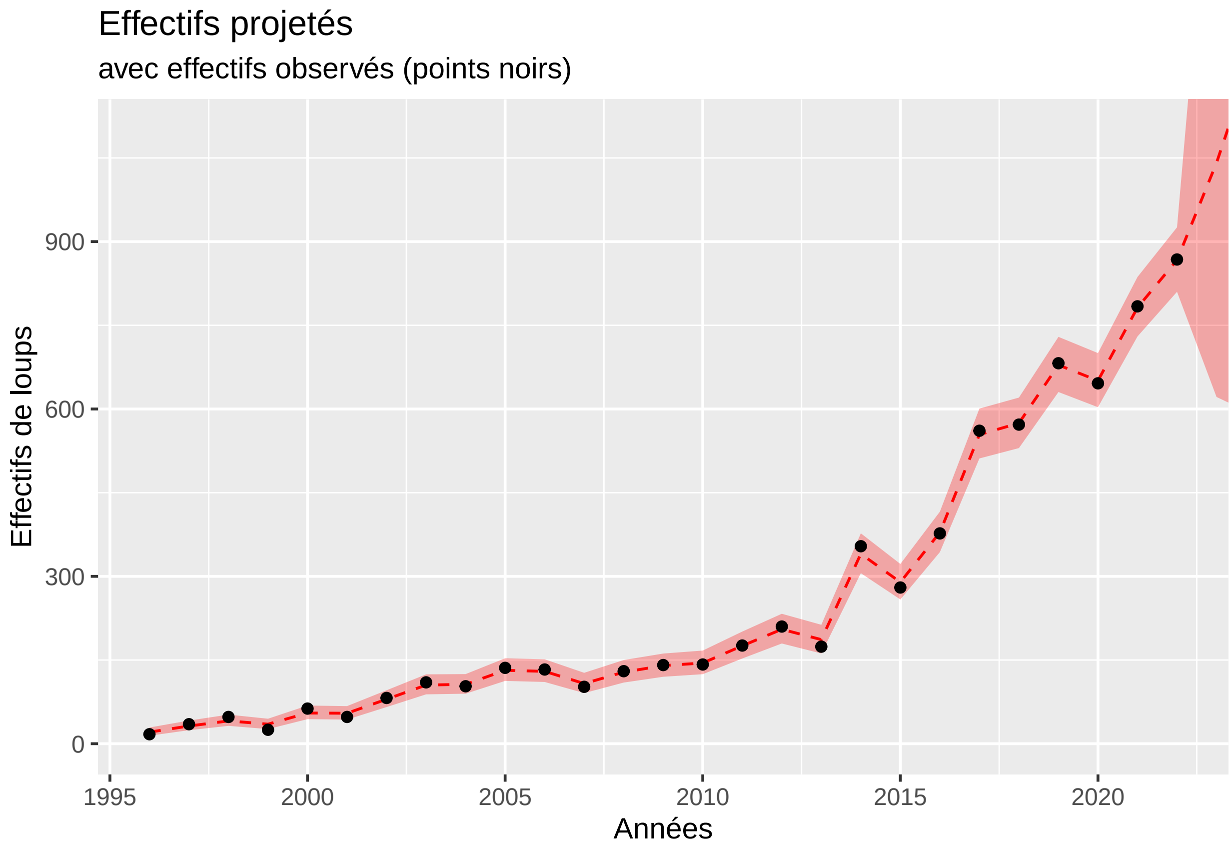
```r
wolf_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
#  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
#  filter(str_detect(parameter, "lambda")) %>%
  ggplot() +
  aes(x = lambda) +
```

```
  geom_density() +
  geom_vline(xintercept = 1, lty = "dashed", color = "red") +
  labs(x = "Taux de croissance")
```



Ensuite les projections.

```
wolf_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
  filter(str_detect(parameter, "N")) %>%
  group_by(parameter) %>%
  summarize(medianN = median(value),
            lci = quantile(value, probs = 2.5/100),
            uci = quantile(value, probs = 97.5/100)) %>%
  mutate(an = parse_number(parameter) + 1995) %>%
  arrange(an) %>%
  ggplot() +
  geom_ribbon(aes(x = an, y = medianN, ymin = lci, ymax = uci), fill = "red", alpha = 0.3) +
  geom_line(aes(x = an, y = medianN), lty = "dashed", color = "red") +
#  geom_point(aes(x = an, y = medianN), color = "red") +
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  coord_cartesian(xlim=c(1996,2022),ylim=c(0,1100))+
  labs(y = "Effectifs de loups",
       x = "Années",
       title = "Effectifs projetés",
       subtitle = "avec effectifs observés (points noirs)")
```



Forecasting

Le modèle décrit l'évolution des effectifs à $t+1$ en fonction des effectifs à t et permet donc de projeter les effectifs en 2021 en connaissant les effectifs de 2020 la dernière année du suivi, puis ceux de 2023 en utilisant les effectifs prédits pour 2022, et ainsi de suite. A chaque étape, il y a des erreurs qui s'accumulent. L'approche bayésienne a l'avantage de permettre de faire ces prédictions en reportant les incertitudes d'une année à l'autre. C'est ce qui fait des modèles à espace d'états en bayésien un outil très utile pour faire des projections.

Bien. Maintenant dans le modèle utilisé, la variable effectifs prélevés est supposée connue. Il s'agit d'une donnée, et par définition on ne la connait pas dans le futur. Il nous faut donc un modèle sur les effectifs prélevés, comme on en a un sur les effectifs comptés.

Andrén et al. proposent le modèle à espace d'états suivant :

$$H_t \sim \text{log-Normale}\left(\max(0, \log(b_0 + b_1 y_{t-1})), \sigma_q^2\right)$$

et

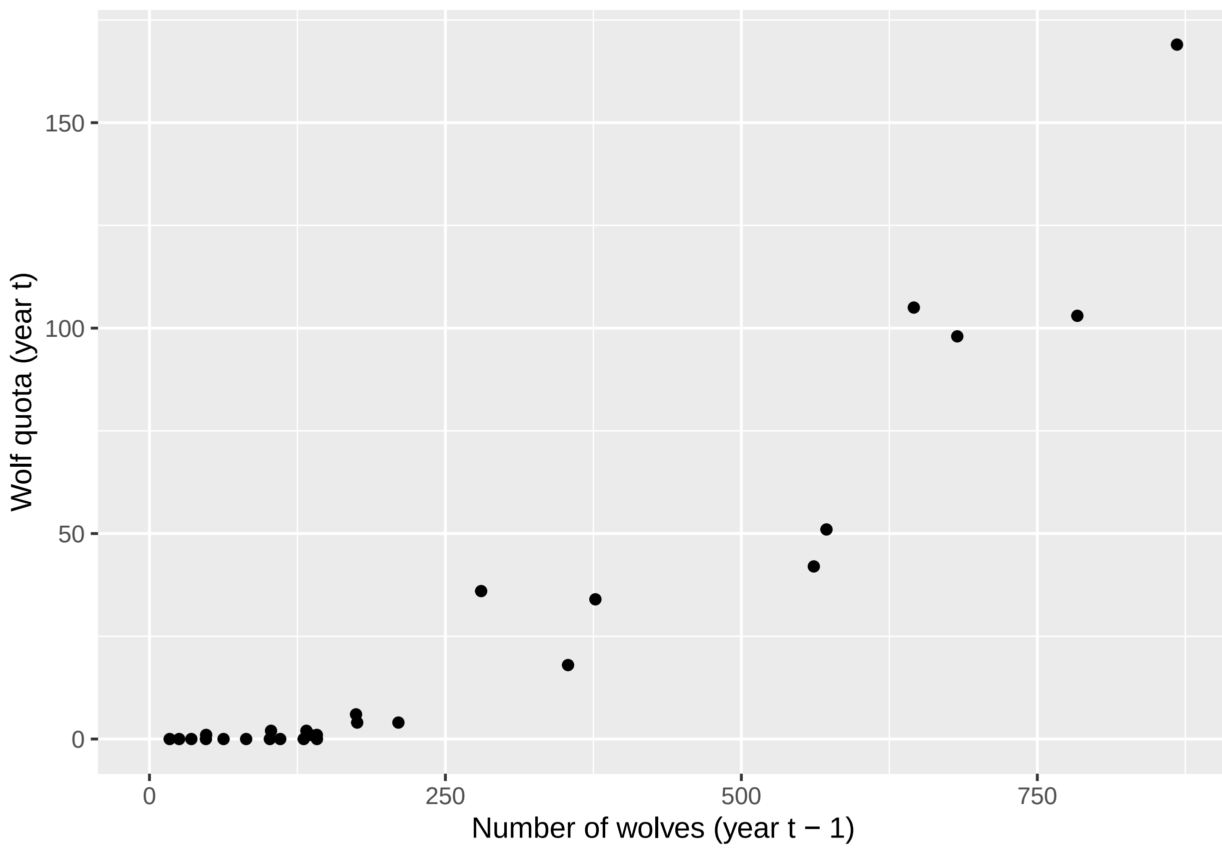
$$q_t \sim \text{Poisson}(H_t)$$

où q_t est le quota observé au temps t et H_t l'effectif réel d'animaux prélevés. La prédiction du modèle est H_t avec une erreur de processus σ_q^2.

On retrouve l'astuce utilisée par Guillaume pour forcer la moyenne de la normale à être supérieure ou égale à 0 avec le $\max(0, \log)$.

On a deux scénarios, ou bien un quota proportionnel aux effectifs comptés avec $b_0 = 0$ (modèle 1 : proportional quota setting strategy), ou bien des prélèvements qui augmentent proportionnellement, avec un quota nul en-dessous d'un seuil (modèle 2 : threshold quota setting strategy). Ce seuil X se calcule en fixant $0 = b_0 + b_1 X$ soit $X = -b_0/b_1$. J'ai pas tout bien compris encore à ce scénario. Ca deviendra plus clair en essayant d'ajuster les modèles je suppose.

```
ggplot() +
  geom_point(aes(x = CMR, y = harvest), color = "black")  +
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of wolves (year t - 1)",
       y = "Wolf quota (year t)")
```



Modèle 1

Commençons par le modèle 1.

```
model1 <- function(){

  # Priors
  sigmaProc ~ dunif(0, 4)
  tauProc <- 1/sigmaProc^2
  b[1] ~ dnorm(0, 3)

  # Process model
  for (t in 1:(nyears)) {
    mu[t] <- log(b[1] * y[t])
    Hproc[t] <- max(0, mu[t])
```

```
    H[t] ~ dlnorm(Hproc[t], tauProc)
    }

  # Observation model
  for (t in 1:nyears) {
    q[t] ~ dpois(H[t])
  }

}
```

On prépare les données.

```
bugs.data <- list(
    nyears = 27,
    y = CMR,
    q = harvest)
```

On précise les paramètres à estimer et le nombre de chaines de MCMC (j'en prends trois ici).

```
bugs.monitor <- c("b", "sigmaProc","H")
bugs.chains <- 3
bugs.inits <- function(){
    list(
    )
}
```

Allez zooh, on lance la machine!

```
mod1 <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model1,
                  n.chains = bugs.chains,
                                                    n.thin = 10,
                                                    n.iter = 100000,
                                                    n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 29
##    Total graph size: 172
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

```
print(mod1, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/tmp/RtmpurfewI/model167f310adecc.txt", fit using jags,
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##            mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## H[1]         0.474   0.534   0.013   0.299   1.941 1.001  4100
## H[2]         0.498   0.557   0.015   0.317   2.056 1.001  6100
## H[3]         0.531   0.599   0.015   0.338   2.139 1.001  5900
## H[4]         0.483   0.542   0.013   0.301   1.982 1.001  8900
## H[5]         0.577   0.626   0.017   0.376   2.302 1.001 15000
## H[6]         1.171   0.949   0.119   0.918   3.695 1.001 15000
## H[7]         0.639   0.674   0.020   0.425   2.509 1.001 15000
## H[8]         0.696   0.714   0.025   0.474   2.628 1.001  4300
## H[9]         2.144   1.353   0.385   1.859   5.541 1.001 15000
## H[10]        1.430   1.074   0.162   1.175   4.162 1.001 15000
## H[11]        2.242   1.387   0.427   1.954   5.682 1.001  6100
## H[12]        0.672   0.691   0.022   0.460   2.540 1.001  8400
## H[13]        0.726   0.721   0.027   0.506   2.711 1.001  5600
## H[14]        1.450   1.092   0.161   1.176   4.231 1.002  2500
## H[15]        0.764   0.764   0.028   0.528   2.837 1.001  8200
## H[16]        4.083   1.947   1.247   3.765   8.761 1.001  9100
## H[17]        4.151   1.939   1.295   3.851   8.693 1.001 13000
## H[18]        5.894   2.340   2.296   5.590  11.312 1.001 14000
## H[19]       17.685   4.186  10.416  17.394  26.885 1.001 15000
## H[20]       35.380   5.934  24.786  35.070  48.051 1.001 15000
## H[21]       33.530   5.765  23.180  33.253  45.710 1.001 15000
## H[22]       41.589   6.413  30.008  41.258  55.263 1.001 15000
## H[23]       50.510   7.094  37.565  50.208  65.458 1.001 15000
## H[24]       97.406   9.832  79.357  96.949 117.755 1.001 15000
## H[25]      104.283  10.127  85.515 103.889 125.088 1.001 15000
## H[26]      102.407  10.124  83.478 102.108 123.157 1.001 15000
## H[27]      168.228  13.015 143.779 167.758 194.443 1.002  3100
## b            0.024   0.030   0.011   0.025   0.048 1.260   310
## sigmaProc    1.717   0.421   1.090   1.652   2.737 1.002  2200
## deviance   102.513   7.284  89.980 101.908 118.475 1.001  3900
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 26.5 and DIC = 129.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Le paramètre b_1 est estimé être :

```
mod1$BUGSoutput$mean$b
```

```
## [1] 0.02436241
```

Graphiquement, on obtient.

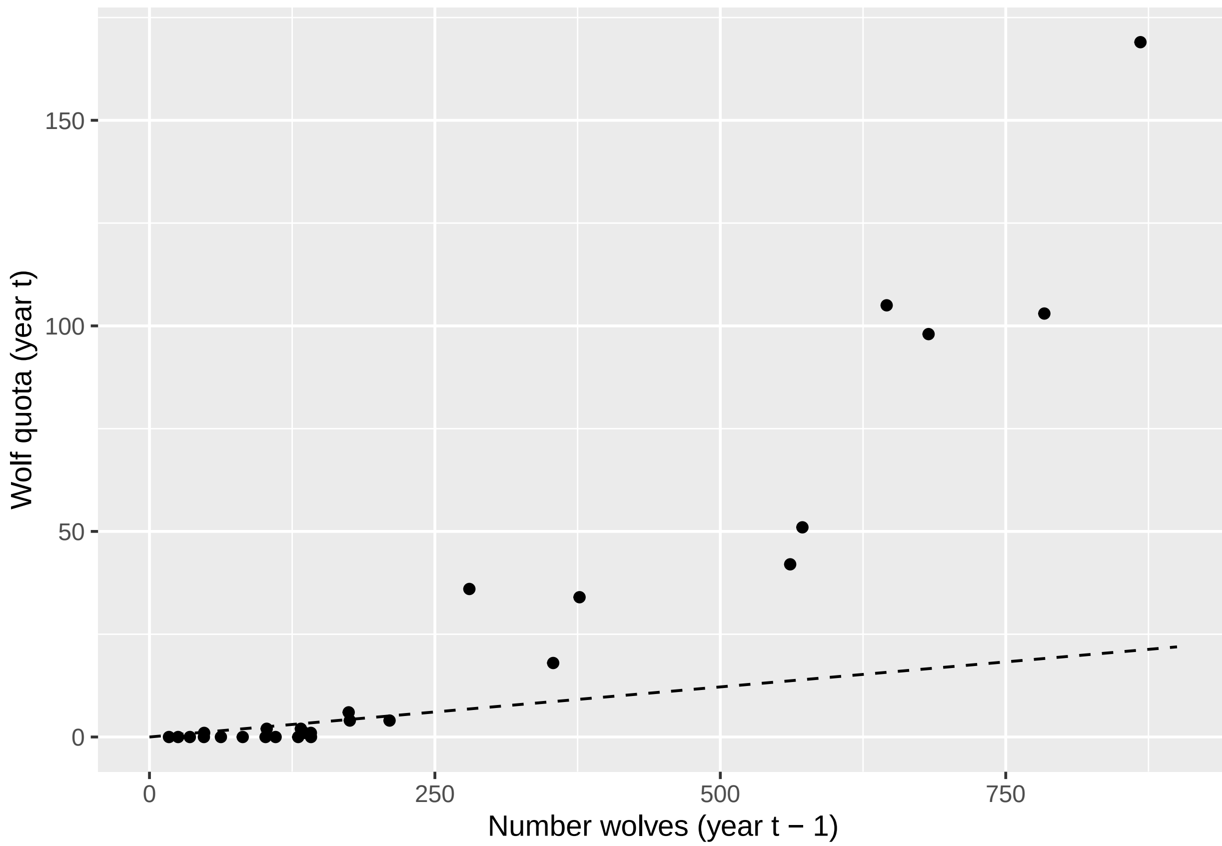
```
grid <- seq(0, 900, length.out = length(CMR))

ggplot() +
```

```r
  geom_point(aes(x = CMR, y = harvest), color = "black") +
  geom_line(aes(x = grid, y = mod1$BUGSoutput$mean$b * grid), color = "black", lty = "dashed") +

  expand_limits(x = 0, y = 0) +
  labs(x = "Number wolves (year t - 1)",
       y = "Wolf quota (year t)")
```

```
## Warning in mod1$BUGSoutput$mean$b * grid: Recycling array of length 1 in array-vector arithmetic is d
##   Use c() or as.vector() instead.
```



Modèle 2

On écrit le modèle. La différence avec le modèe 1 est qu'on estime une ordonnée à l'origine.

```r
model2 <- function(){

  # Priors
  sigmaProc ~ dunif(0, 4)
  tauProc <- 1/sigmaProc^2
  b[1] ~ dnorm(0, 1/3000)
  b[2] ~ dnorm(0, 1/3000)
  # Process model
  for (t in 1:(nyears)) {
    mu[t] <- log(b[1] + b[2] * y[t])
```

```
#    mu[t] <- log(b[1] + b[2] * y[t]) * index[t]
#    index[t] <- - 1000 * step(y[t] + b[1] / b[2]) # step(x) = 1 if x >= 0
#    index[t] <- step(q[t]) # step(x) = 1 if x >= 0
#    mu[t] <- log(b[1] + b[2] * y[t])
    Hproc[t] <- max(0, mu[t])
    H[t] ~ dlnorm(Hproc[t], tauProc)

# les lignes de code suivantes donnent un ajustement pas mal, mais
# sauf qu'à l'approche de census == 0 on a harvest == 0
#    Hproc[t] <- log(b[1] + b[2] * y[t])
#    H[t] ~ dlnorm(Hproc[t], tauProc)
    }

  # Observation model
  for (t in 1:nyears) {
    q[t] ~ dpois(H[t])
  }

}
```

On prépare les données pour la Suède.

```
bugs.data <- list(
    nyears = 27,
    y = CMR,
    q = harvest)
```

On précise les paramètres à estimer et le nombre de chaines de MCMC (j'en prends trois ici).

```
bugs.monitor <- c("b", "sigmaProc")
bugs.chains <- 3
bugs.inits <- function(){
    list(
    )
}
```

Allez zooh, on lance la machine!

```
mod2 <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model2,
                  n.chains = bugs.chains,
                                                    n.thin = 10,
                                                    n.iter = 100000,
                                                    n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
```

```
##    Unobserved stochastic nodes: 30
##    Total graph size: 201
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

```r
print(mod2, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/tmp/RtmpurfewI/model167f2e66708d.txt", fit using jags,
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##           mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## b[1]      -24.178   5.414 -35.887 -23.576 -16.093 1.011  1900
## b[2]        0.162   0.026   0.118   0.159   0.221 1.001  3700
## sigmaProc   0.394   0.127   0.213   0.372   0.706 1.001  5100
## deviance  107.781   5.837  97.684 107.282 120.709 1.001  6200
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 17.0 and DIC = 124.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Les paramètres b sont estimés comme suit.

```r
mod2$BUGSoutput$mean$b
```

```
## [1] -24.1776185   0.1618312
```

Le ratio se calcule comme suit.

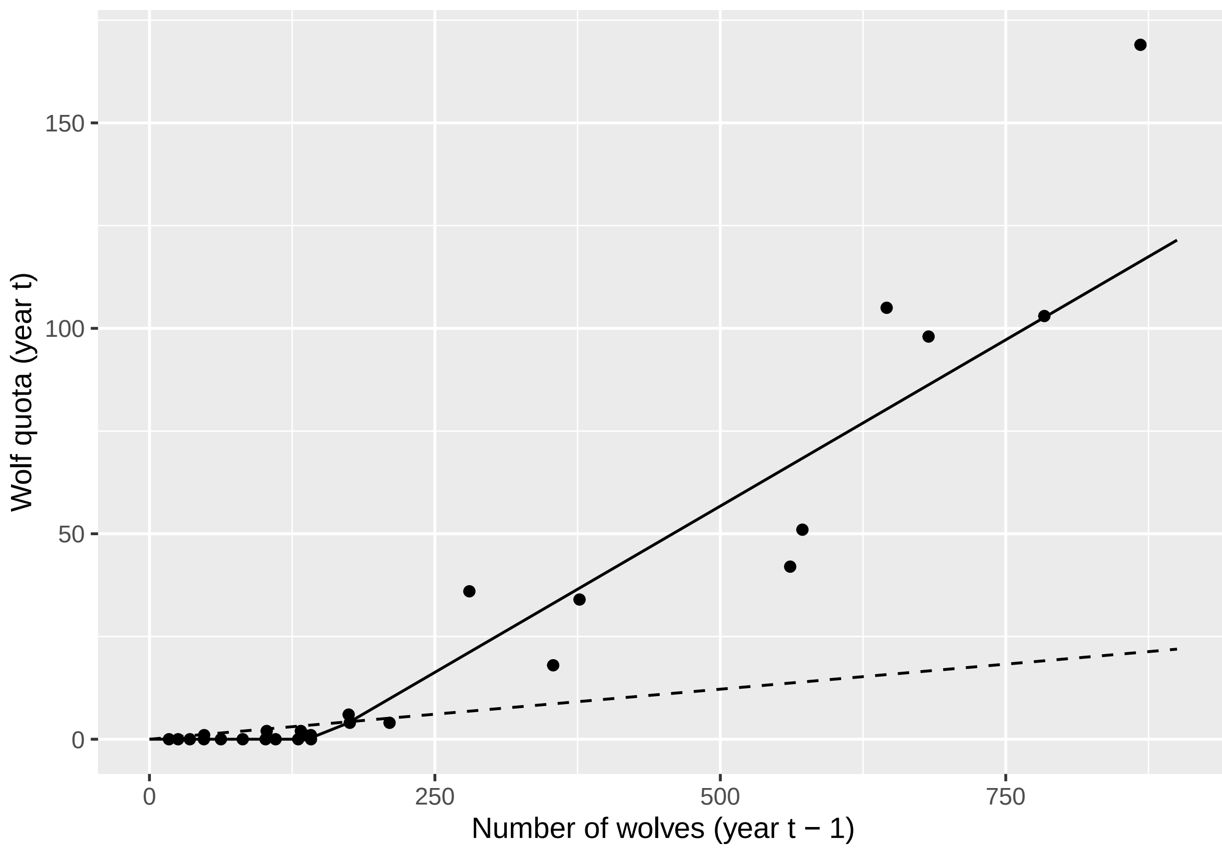
```r
- mod2$BUGSoutput$mean$b[1] / mod2$BUGSoutput$mean$b[2]
```

```
## [1] 149.4002
```

Graphiquement, on obtient.

```r
grid <- seq(0, 900, length.out = length(CMR))
treshold = - mod2$BUGSoutput$mean$b[1] / mod2$BUGSoutput$mean$b[2]
ggplot() +
  geom_point(aes(x = CMR, y = harvest), color = "black") +
  geom_line(aes(x = grid, y = mod1$BUGSoutput$mean$b * grid), color = "black", lty = "dashed") +
  geom_line(aes(x = grid, y = if_else(grid < treshold, 0, (mod2$BUGSoutput$mean$b[1] +  mod2$BUGSoutput
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of wolves (year t - 1)",
       y = "Wolf quota (year t)")
```

```
## Warning in mod1$BUGSoutput$mean$b * grid: Recycling array of length 1 in array-vector arithmetic is
##   Use c() or as.vector() instead.
```



Conclusion

On constate que le meilleur scénario pour déterminer le quota du nombre de loups à tuer est le modèle à seuil (ici à 149).

Exemple de nombre de loup à tuer pour une population de 868 individus.

```
HWolf=function(n){
  if (n>149){
  return(as.integer(-24+0.1618981*n))}
  else {return(0)}}


HWolf(868)
```

```
## [1] 116
```

Amélioration de la projection

A l'aide de l'estimation du nombre de loups à tuer selon la taille de la population, on modélise la projection des effecitfs en tenant compte des prélèvements.

```
modelh <- function(){

  # Priors

  sigmaProc ~ dunif(0, 10)
  tauProc <- 1/sigmaProc^2
  lambda ~ dunif(0, 5)

  N[1] ~ dgamma(1.0E-6, 1.0E-6)

  # Process model
  for (t in 2:(nyears)) {
    mu[t] <- lambda * (N[t-1] - harvest[t-1])
    Nproc[t] <- log(max(1, mu[t]))
    N[t] ~ dlnorm(Nproc[t], tauProc)
  }

  # Observation model
  for (t in 1:nyears) {
    y[t] ~ dpois(N[t])
  }

  # Projected population
  for (t in (nyears + 1):(nyears + 10)) {
    mu[t] <- lambda * (N[t-1] - (-24+0.1618981*N[t-1]))
    Nproc[t] <- log(max(1, mu[t]))
    N[t] ~ dlnorm(Nproc[t], tauProc)
  }
}
```

On prépare les données.

```
bugs.data <- list(
    nyears = nrow(thedata),
    y = round(thedata$N),
    harvest = thedata$H)
```

On précise les paramètres à estimer et le nombre de chaines de MCMC (j'en prends trois ici).

```
bugs.monitor <- c("lambda", "sigmaProc","N", "tauProc")
bugs.chains <- 3
bugs.inits <- function(){
    list(
    )
}
```

Allez zooh, on lance la machine!

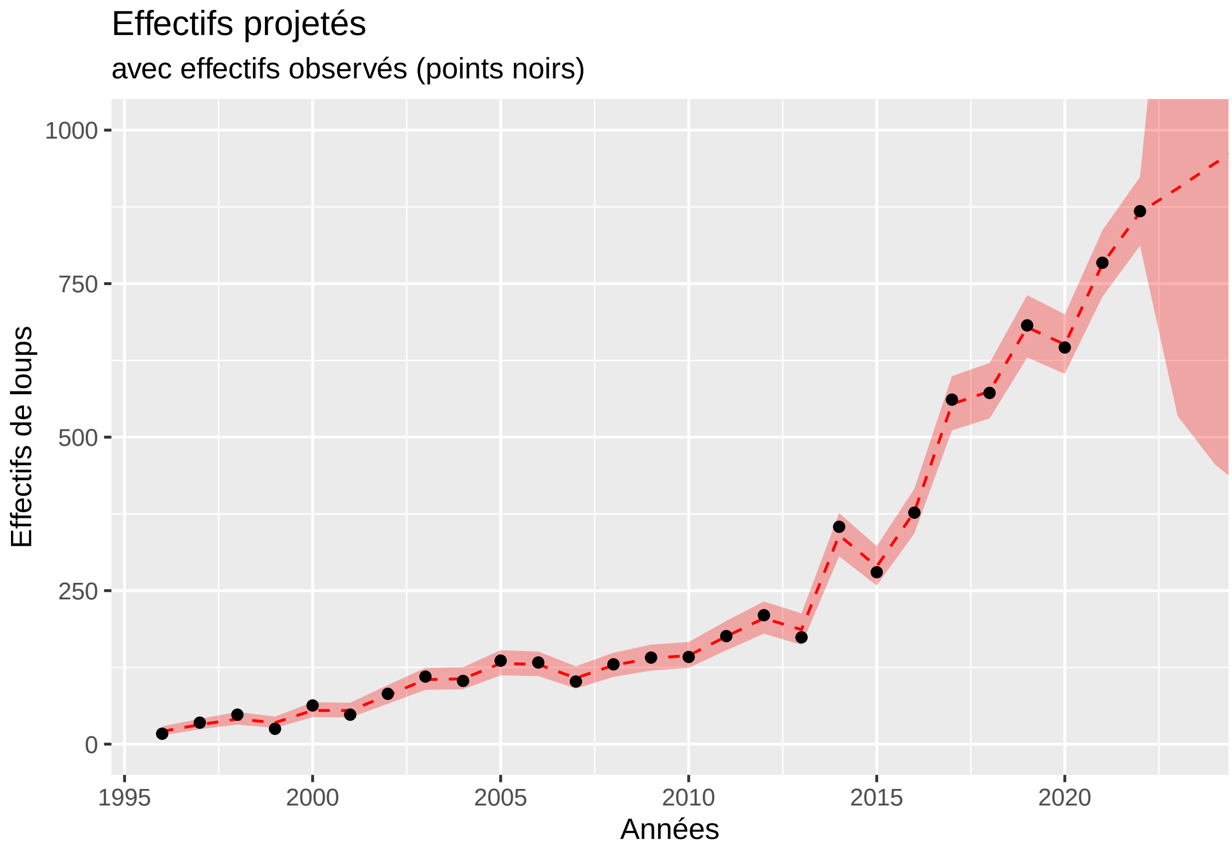
```
library(R2jags)
wolf_modh <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
```

```
                  model.file = modelh,
                  n.chains = bugs.chains,
                                                    n.thin = 10,
                                                    n.iter = 100000,
                                                    n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 39
##    Total graph size: 269
##
## Initializing model
```

Ensuite les projections.

```
wolf_modh$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
  filter(str_detect(parameter, "N")) %>%
  group_by(parameter) %>%
  summarize(medianNh = median(value),
            lci = quantile(value, probs = 2.5/100),
            uci = quantile(value, probs = 97.5/100)) %>%
  mutate(an = parse_number(parameter) + 1995) %>%
  arrange(an) %>%
  ggplot() +
  geom_ribbon(aes(x = an, y = medianNh, ymin = lci, ymax = uci), fill = "red", alpha = 0.3) +
  geom_line(aes(x = an, y = medianNh), lty = "dashed", color = "red") +
#  geom_point(aes(x = an, y = medianN), color = "red") +
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  coord_cartesian(xlim=c(1996,2023),ylim=c(0,1000))+
  labs(y = "Effectifs de loups",
       x = "Années",
       title = "Effectifs projetés",
       subtitle = "avec effectifs observés (points noirs)")
```


Effectifs projetés
avec effectifs observés (points noirs)
Effectifs de loups
Années
0
250
500
750
1000
1995
2000
2005
2010
2015
2020