Modèle de gestion adaptative du loup

Olivier Gimenez & Loïc Pages

12/01/2024

Introduction

Nous allons ici reprendre différents modèles d'estimation de population : le modèle exponentiel et le modèle logistique. Ces modèles seront appliqués à la population de loups en France. Nous allons également y ajouter un cadre prédictionnel dans une optique de gestion adaptative sur un intervalle de temps de 2 ans. L'efficacité des deux modèles sera comparée par le DIC.

Les modèle exponentiel d'estimation utilisé dans ce code provient de l'article de Andrén et al.

Dans un dernier temps, nous simulerons des données à l'aide des paramètres estimés afin de voir si les estimations collent avec tous types de données. Puis nous pourrons faire une projection sur 20 ans avec les deux types de modèle.

Préparation

```r
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

Estimations d'effectifs par CMR :

```
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)
```

Nombre de prélèvements :

```
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)
```

Erreur d'observation :

```
ObsSE=rep(0.3,27)
# se = read_csv("se.csv") %>%
#   as_tibble()
```

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

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

```
##      N  se  H
## 1   17 0.3  0
## 2   35 0.3  0
## 3   48 0.3  0
## 4   25 0.3  0
## 5   63 0.3  0
## 6   48 0.3  1
## 7   82 0.3  0
## 8  110 0.3  0
## 9  103 0.3  2
## 10 136 0.3  1
## 11 133 0.3  2
## 12 102 0.3  0
## 13 130 0.3  0
## 14 141 0.3  1
## 15 142 0.3  0
## 16 176 0.3  4
## 17 210 0.3  4
## 18 174 0.3  6
## 19 354 0.3 18
## 20 280 0.3 36
## 21 377 0.3 34
```

```
## 22 561 0.3  42
## 23 572 0.3  51
## 24 682 0.3  98
## 25 646 0.3 105
## 26 784 0.3 103
## 27 868 0.3 169
```

Modèles d'estimation et de prédiction à cours terme

Modèle exponentiel

Dans ce modèle, l'effectif de la population suit une croissance exponentielle. On soustrait le nombre de prélèvement à l'effectif de la population au temps $t-1$ puis on le multiplie par le taux de reproduction λ. On obtient l'effectif de la population au temps t.

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

On ajoute à cette relation déterministe de la stochasticité. Ici l'effectif de la population au temps t suit un loi log-normale, c'est à dire que les effectifs sont normalement distribués sur l'échelle log :

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

avec la moyenne $\mu_t = \log(N_t) = \log(\lambda(N_{t-1} - H_{t-1}))$ et σ_{proc} l'erreur standard des effectifs. On utilise une loi log-normale plutôt qu'une loi de Poisson car les estimations semblent être plus précises et suivent mieux les données observées.

On ajoute les effectifs observés y_t qui suivent une loi de Poisson de paramètre l'effectif estimé au temps t.

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

On modélise tout ça en bayésien :

```
modelexp = 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] - h[t-1])
    NProc[t] = log(max(1, mu[t]))
    N[t] ~ dlnorm(NProc[t], tauProc)
   # N[t] ~ dpois(mu[t])
  }

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

Initialisation des données :

```
bugs.data = list(nyears = nrow(dat),
                 y = dat$N,
                 h = dat$H,
                 yse=dat$se)
```

Paramètres JAGS :

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

Lancement du modèle.

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

```
## module glm loaded
```

```
## Warning in jags.model(model.file, data = data, inits = init.values, n.chains =
## n.chains, : Unused variable "yse" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 29
##    Total graph size: 196
##
## Initializing model
```

On affiche les estimations obtenus.

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

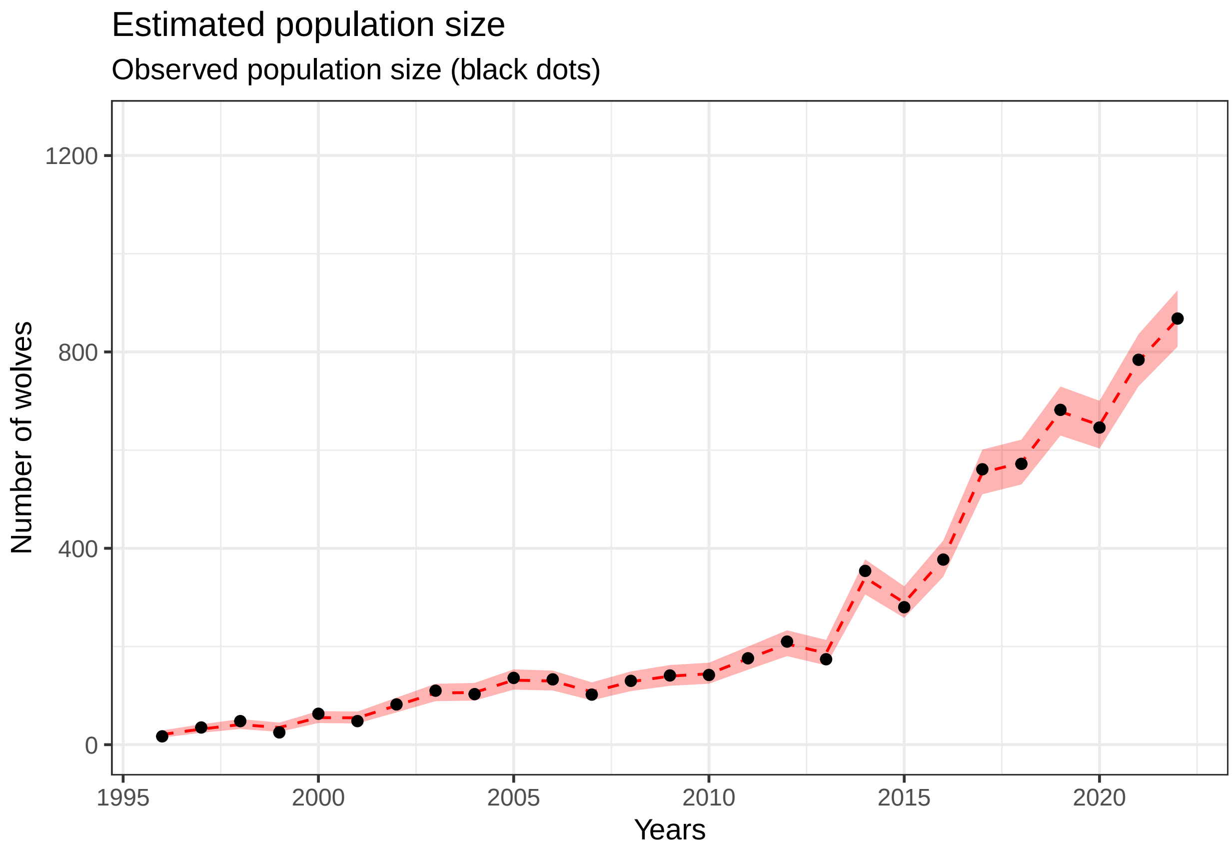
```
## Inference for Bugs model at "/tmp/RtmpezfqFq/model2df947dd5e9.txt", fit using jags,
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##           mu.vect sd.vect   2.5%     50%   97.5% Rhat n.eff
## N[1]       20.997   3.757  14.306  20.762  29.033 1.001 15000
## N[2]       32.052   4.410  24.207  31.766  41.558 1.001 15000
```

```
## N[3]       41.224   5.167  31.888  40.948  52.147 1.001 15000
## N[4]       35.164   4.794  26.142  35.016  44.972 1.001 13000
## N[5]       55.444   6.249  44.190  55.098  68.480 1.001  7800
## N[6]       54.883   6.177  43.167  54.727  67.497 1.001 13000
## N[7]       80.007   7.675  65.812  79.736  95.764 1.001 15000
## N[8]      105.411   9.064  88.668 105.196 124.075 1.001 15000
## N[9]      106.683   9.145  89.730 106.347 125.780 1.001 11000
## N[10]     131.796  10.499 112.141 131.419 153.408 1.001 15000
## N[11]     130.070  10.378 110.403 129.742 151.002 1.001  7700
## N[12]     107.874   9.360  90.345 107.664 126.934 1.001 15000
## N[13]     128.396  10.218 109.164 128.159 149.210 1.001 15000
## N[14]     140.077  10.720 119.876 139.692 162.239 1.001 15000
## N[15]     144.674  10.848 124.178 144.380 166.980 1.001 15000
## N[16]     175.615  12.158 152.774 175.280 199.828 1.001 15000
## N[17]     205.529  13.429 180.342 205.105 233.103 1.001 13000
## N[18]     186.743  13.303 161.683 186.504 213.383 1.001 15000
## N[19]     340.478  18.125 306.002 340.210 377.416 1.001 15000
## N[20]     289.600  16.396 258.255 289.299 322.786 1.001 15000
## N[21]     378.767  18.735 342.033 378.541 416.044 1.001  6500
## N[22]     554.582  22.981 510.087 554.044 601.060 1.001 15000
## N[23]     574.746  23.084 530.036 574.573 621.365 1.001  9000
## N[24]     678.806  25.442 629.691 678.431 729.687 1.001 15000
## N[25]     651.195  24.592 603.319 651.011 700.474 1.001 15000
## N[26]     781.737  27.077 730.037 781.305 836.202 1.001 15000
## N[27]     867.262  29.359 810.773 866.888 925.603 1.001  8300
## lambda      1.208   0.062   1.092   1.206   1.338 1.001 15000
## sigmaProc   0.250   0.053   0.164   0.244   0.371 1.001 15000
## tauProc    18.268   7.755   7.282  16.845  37.283 1.001 15000
## deviance  218.821   8.234 204.655 218.285 236.552 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 = 33.9 and DIC = 252.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

On affiche la dynamique de la population sur un graphique.

```r
wolf_modelexp$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
            hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter) + 1995)%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
```

```
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  coord_cartesian(xlim=c(1996,2022),ylim=c(0,1250))+
  theme_bw()+
  labs(title = "Estimated population size",
       subtitle = "Observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```



Projection

On va maintenant ajouter une projection sur 2 ans pour différents taux de prélèvement :

```
dH = c(0, 0.10, 0.20, 0.30)
```

Le modèle est le même que précédemment à l'exeption de la partie Projected model qui ajoute les prédicitions au modèle.

```
modelexp = 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] - h[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 model
  for (t in (nyears + 1):(nyears + 2)) {
    mu[t] = (lambda - dH) * N[t - 1]
    NProc[t] = log(max(1, mu[t]))
    N[t] ~ dlnorm(NProc[t], tauProc)
  }
}
```

On lance la machine pour chaque taux de prélevement.

```
for (i in 1:4) {
  # Initialisation des données
  bugs.data = list(
    nyears = nrow(dat),
    y = c(dat$N, rep(NA, 2)),
    dH = dH[i],
    h = dat$H
  )

  # Paramètres jags
  bugs.monitor = c("lambda", "sigmaProc", "N", "tauProc")
  bugs.chains = 3
  bugs.inits = function() {
    list()
  }

# Lancement du modèle

wolf_modelexp = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modelexp,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=100000,
                   n.burnin=50000)

if (i==1){
output1 = wolf_modelexp$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter1") %>%
```

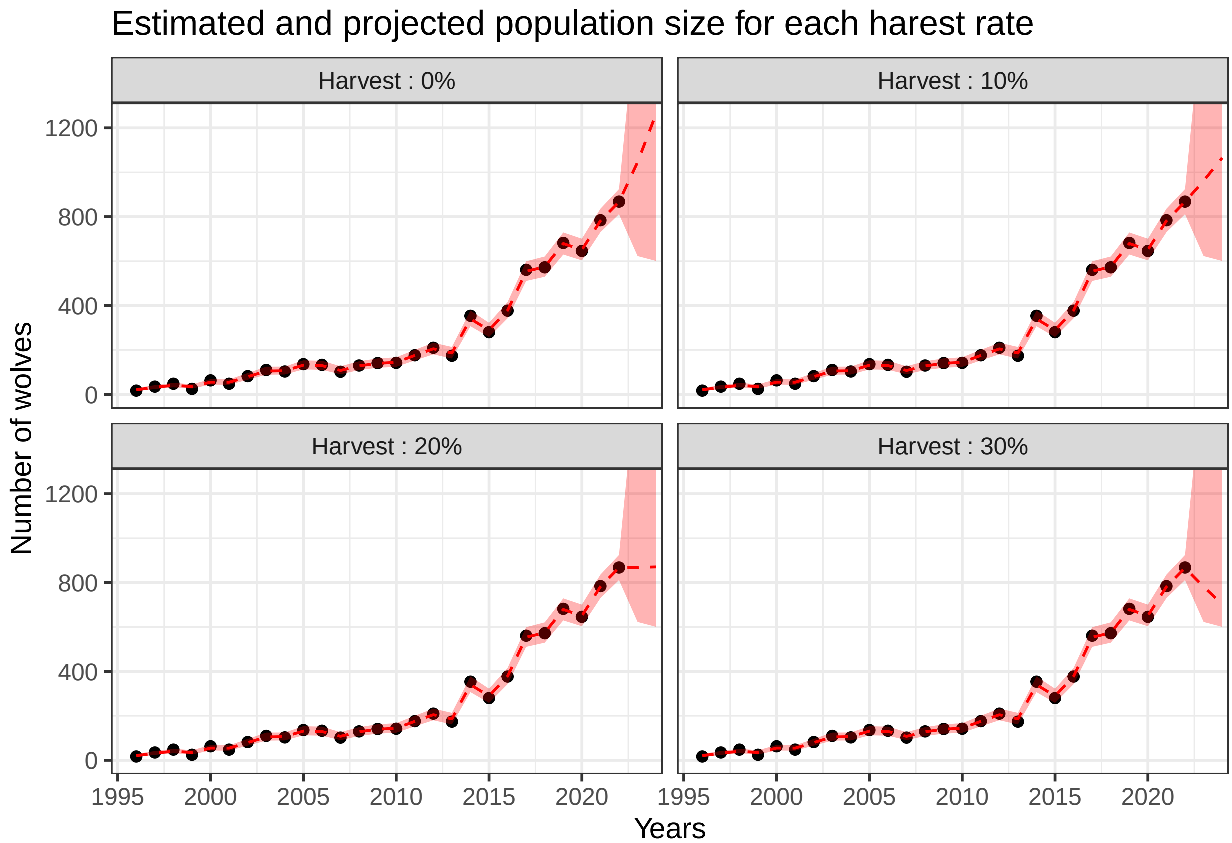
```
  filter(str_detect(parameter1, "N")) %>%
  group_by(parameter1) %>%
  summarize(medianN1 = median(value),
            lq1 = quantile(value, probs = 2.5/100),
            hq1 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter1) + 1995)%>%
  arrange(years)%>%
  mutate(ObsY = bugs.data$y)
}

if(i==2){
  output2 = wolf_modelexp$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter2") %>%
  filter(str_detect(parameter2, "N")) %>%
  group_by(parameter2) %>%
  summarize(medianN2 = median(value),
            lq2 = quantile(value, probs = 2.5/100),
            hq2 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter2) + 1995)%>%
  arrange(years)
}

if(i==3){
  output3 = wolf_modelexp$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter3") %>%
  filter(str_detect(parameter3, "N")) %>%
  group_by(parameter3) %>%
  summarize(medianN3 = median(value),
            lq3 = quantile(value, probs = 2.5/100),
            hq3 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter3) + 1995)%>%
  arrange(years)
}

if(i==4){
  output4 = wolf_modelexp$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter4") %>%
  filter(str_detect(parameter4, "N")) %>%
  group_by(parameter4) %>%
  summarize(medianN4 = median(value),
            lq4 = quantile(value, probs = 2.5/100),
            hq4 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter4) + 1995)%>%
  arrange(years)
}
}
```

On affiche les courbes d'effectifs :

```r
output = output1 %>% left_join(output2) %>%
  left_join(output3) %>%
  left_join(output4) %>%
  pivot_longer(
    c(medianN1, medianN2, medianN3, medianN4),
    names_to = "medianN",
    values_to = "valuesM")
```

```
## Joining with `by = join_by(years)`
## Joining with `by = join_by(years)`
## Joining with `by = join_by(years)`
```

```r
variable_names <- list(
  "medianN1" = "Harvest : 0%" ,
  "medianN2" = "Harvest : 10%",
  "medianN3" = "Harvest : 20%",
  "medianN4" = "Harvest : 30%")

variable_labeller <- function(variable, value) {
  return(variable_names[value])
}

  ggplot(output)+
  geom_point(aes(x = years, y = ObsY)) +
  coord_cartesian(xlim=c(1996,2023),ylim=c(0,1250))+
  aes(x = years, y = valuesM)+
  geom_line(colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq1, ymax = hq1), fill = "red", alpha = 0.3)+
  facet_wrap(~medianN,labeller = variable_labeller)+
  theme_bw()+
  labs(title = "Estimated and projected population size for each harest rate",
       x = "Years",
       y = "Number of wolves")
```



On définit un objectif d'un maximum d'effectifs à 1250, et un minimum à 1000. Pour atteindre cet objectif on peut imposer un taux de prélèvement de 0% ou 10% sur 2 ans.

Modèle logistique

On définit ici d'abordDans ce modèle, l'effectif de la population suit une croissance logistique, c'est à dire que la population croit de manière exponentielle puis est limitée par une capacité de charge. On soustrait le nombre de prélevements à l'effectif de la population au temps $t-1$. Puis en utilisant ce résultat on calcule

$$\lambda_t = N_{t-1} \times \exp(\alpha(1 - \frac{N_{t-1}}{K})$$

, avec K la capactié de charge.

On ajoute à cette relation déterministe de la stochasticité. L'effectif de la population au temps t suit une loi log-normale, c'est à dire que les effectifs sont normalement distribués sur l'échelle log :

$$log(N_t) \sim \text{Normale}(log(\lambda_{t-1}), \sigma_{\text{proc}})$$

avec σ_{proc} l'erreur standard des effectifs.

On ajoute les effectifs observés yt qui suivent une loi de Poisson de paramètre l'effectif estimé au temps t.

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

Ce qui donne en bayésien :

```r
modellogist = function() {
  # Priors
  sigmaProc ~ dunif (0, 10)
  tauProc = 1 / sigmaProc ^ 2
  alpha ~ dunif(0, 1.0986) #maximum exponential growth rate
  K ~ dunif(1, 1000)         #carrying capacity

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

  # Process model
  for (t in 2:(nyears)) {
    u[t-1] = N[t-1] - h[t-1]
    Er[t] = exp(alpha * (1 - u[t-1] / K)) # per capita growth rate is density dependent - Ricker model
    lambda[t] = u[t-1] * Er[t]
    NProc[t] = log(max(1, lambda[t]))
    N[t] ~ dlnorm(NProc[t], tauProc)
  }
  # Observation model
  for (t in 1:(nyears)) {
    y[t] ~ dpois(N[t])
  }
}
```

Initialisation des données :

```r
bugs.data = list(nyears = nrow(dat),
                 y = dat$N,
                 h = dat$H)
```

Paramètres JAGS :

```r
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)
```

Lancement du modèle.

```r
wolf_modellogist = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modellogist,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=50000,
                   n.burnin=10000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

```
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 30
##    Total graph size: 302
##
## Initializing model
```

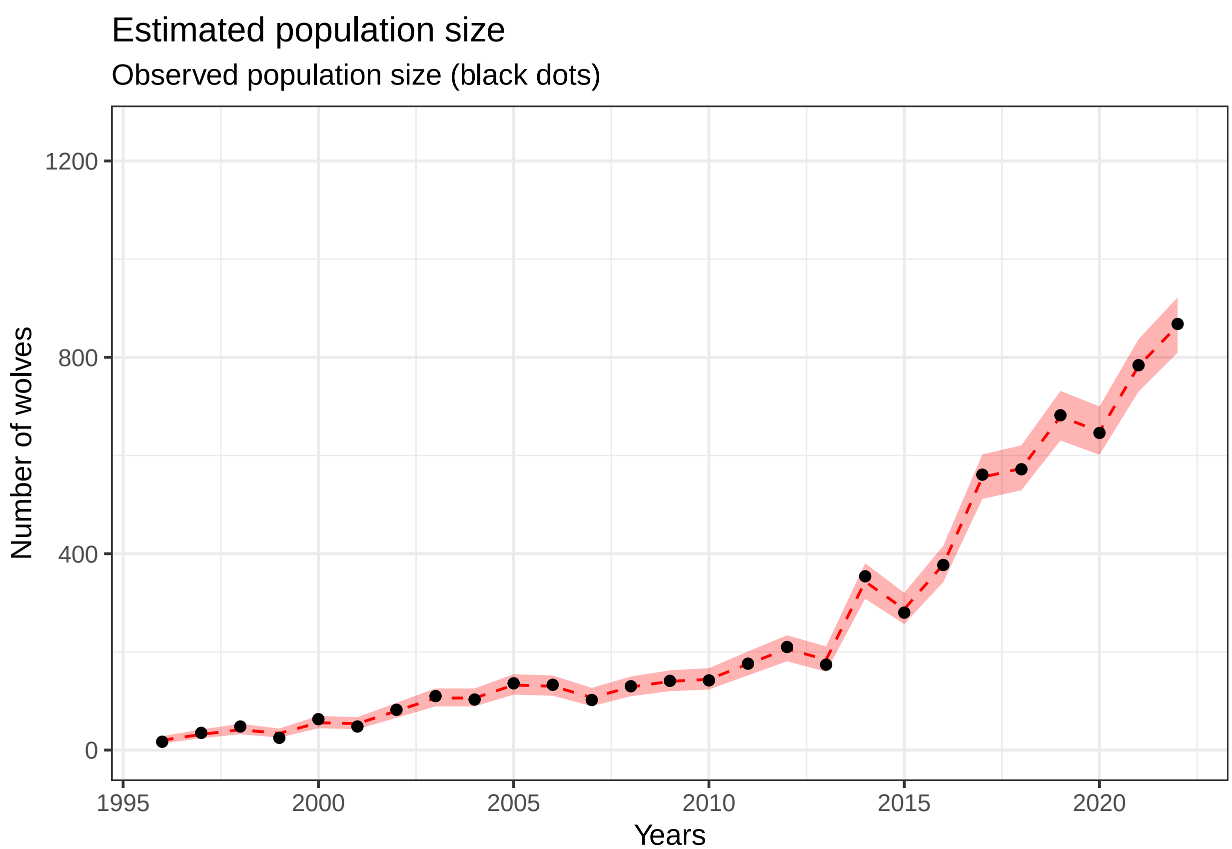
On affiche les estimations obtenus.

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

```
## Inference for Bugs model at "/tmp/RtmpezfqFq/model2df94c8ab5bb.txt", fit using jags,
##  3 chains, each with 50000 iterations (first 10000 discarded), n.thin = 10
##  n.sims = 12000 iterations saved
##           mu.vect sd.vect   2.5%     50%   97.5%  Rhat n.eff
## K         782.998 153.640 445.357 808.868 991.030 1.001  9800
## N[1]       20.283   3.805  13.546  20.052  28.359 1.001 12000
## N[2]       32.210   4.572  24.059  31.923  42.108 1.001 12000
## N[3]       41.924   5.481  32.263  41.594  53.425 1.001 12000
## N[4]       34.279   4.807  25.344  34.078  44.054 1.001 12000
## N[5]       56.082   6.495  44.439  55.704  69.679 1.001 12000
## N[6]       54.120   6.300  42.482  53.878  67.031 1.001 12000
## N[7]       80.167   7.868  65.654  79.765  96.507 1.002  2700
## N[8]      106.277   9.379  89.086 105.997 125.656 1.001  7100
## N[9]      106.343   9.282  88.958 105.993 125.546 1.001  6600
## N[10]     132.763  10.589 112.920 132.442 154.570 1.001 12000
## N[11]     130.504  10.513 110.741 130.151 151.968 1.001 12000
## N[12]     107.291   9.512  89.515 107.028 126.665 1.001  6400
## N[13]     128.761  10.304 109.659 128.436 149.871 1.001 12000
## N[14]     140.327  10.897 119.951 140.053 162.516 1.001 12000
## N[15]     144.423  11.169 123.388 144.108 166.948 1.001 12000
## N[16]     175.793  12.504 152.217 175.439 200.975 1.001 12000
## N[17]     206.257  13.548 180.800 206.046 233.887 1.001 12000
## N[18]     184.467  13.133 159.704 184.132 211.185 1.001 12000
## N[19]     343.488  18.385 308.256 343.170 380.557 1.001 12000
## N[20]     287.391  16.319 256.583 286.987 320.419 1.001  9200
## N[21]     378.373  18.882 342.194 378.095 416.200 1.001 12000
## N[22]     556.264  23.139 511.481 555.953 602.192 1.001 12000
## N[23]     573.952  23.387 529.310 573.533 620.766 1.001  3900
## N[24]     679.834  25.614 630.729 679.309 731.516 1.002  1400
## N[25]     649.286  25.145 601.393 648.955 700.047 1.001 12000
## N[26]     782.604  27.377 730.406 782.120 836.990 1.001 12000
## N[27]     864.816  28.901 809.514 864.684 922.335 1.001 12000
## alpha       0.225   0.074   0.078   0.224   0.374 1.001 12000
## sigmaProc   0.277   0.056   0.187   0.271   0.405 1.001 11000
## tauProc    14.592   5.781   6.096  13.620  28.558 1.001 11000
## deviance  217.197   7.887 203.667 216.517 234.101 1.001 12000
##
## 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 = 31.1 and DIC = 248.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

On affiche la dynamique de la population sur un graphique.

```
wolf_modellogist$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
            hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter) + 1995)%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = dat$N)) +
  coord_cartesian(xlim=c(1996,2022),ylim=c(0,1250))+
  theme_bw()+
  labs(title = "Estimated population size",
       subtitle = "Observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```



Projection

On va maintenant ajouter une projection sur 2 ans pour différents taux de prélèvement :

```
dH = c(0, 0.10, 0.20, 0.30)
```

Le modèle est le même que précédemment à l'exeption de la partie Projected model qui ajoute les prédicitions au modèle.

```
modellogist = function() {
  # Priors
  sigmaProc ~ dunif (0, 5)
  tauProc = 1 / sigmaProc ^ 2
  alpha ~ dunif(0, 1.0986) #maximum exponential growth rate
  K ~ dunif(1, 1000)         #carrying capacity

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

  # Process model
  for (t in 2:(nyears)) {
    u[t-1] = N[t-1] - h[t-1]
    Er[t] = exp(alpha * (1 - u[t-1] / K)) # per capita growth rate is density dependent - Ricker model
    lambda[t] = u[t-1] * Er[t]
    NProc[t] = log(max(1, lambda[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)) {
    u[t-1] = (1-dH) * N[t-1]
    Er[t] = exp(alpha * (1 - u[t-1] / K)) # per capita growth rate is density dependent - Ricker model
    lambda[t] = u[t-1] * Er[t]
    NProc[t] = log(max(1, lambda[t]))
    N[t] ~ dlnorm(NProc[t], tauProc)
  }
}
```

On lance la machine pour chaque taux et on affiche la courbe d'effectifs :

```
for (i in 1:4) {
  # Initialisation des données
  bugs.data = list(
    nyears = nrow(dat),
    y = c(dat$N, rep(NA, 2)),
    dH = dH[i],
    h = dat$H
  )

  # Paramètres jags
  bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "N")
```

```
  bugs.chains = 3
  init1 = list(alpha = .5, sigmaProc = .25)
  init2 = list(alpha = .1, sigmaProc = .05)
  init3 = list(alpha = 1, sigmaProc = .45)
  bugs.inits = list(init1, init2, init3)

  # Lancement du modèle

wolf_modellogist = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modellogist,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=100000,
                   n.burnin=50000)

if (i==1){
output1 = wolf_modellogist$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter1") %>%
  filter(str_detect(parameter1, "N")) %>%
  group_by(parameter1) %>%
  summarize(medianN1 = median(value),
            lq1 = quantile(value, probs = 2.5/100),
            hq1 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter1) + 1995)%>%
  arrange(years)%>%
  mutate(ObsY = bugs.data$y)
}

if(i==2){
  output2 = wolf_modellogist$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter2") %>%
  filter(str_detect(parameter2, "N")) %>%
  group_by(parameter2) %>%
  summarize(medianN2 = median(value),
            lq2 = quantile(value, probs = 2.5/100),
            hq2 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter2) + 1995)%>%
  arrange(years)
}

if(i==3){
  output3 = wolf_modellogist$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter3") %>%
  filter(str_detect(parameter3, "N")) %>%
  group_by(parameter3) %>%
  summarize(medianN3 = median(value),
            lq3 = quantile(value, probs = 2.5/100),
            hq3 = quantile(value, probs = 97.5/100))%>%
```

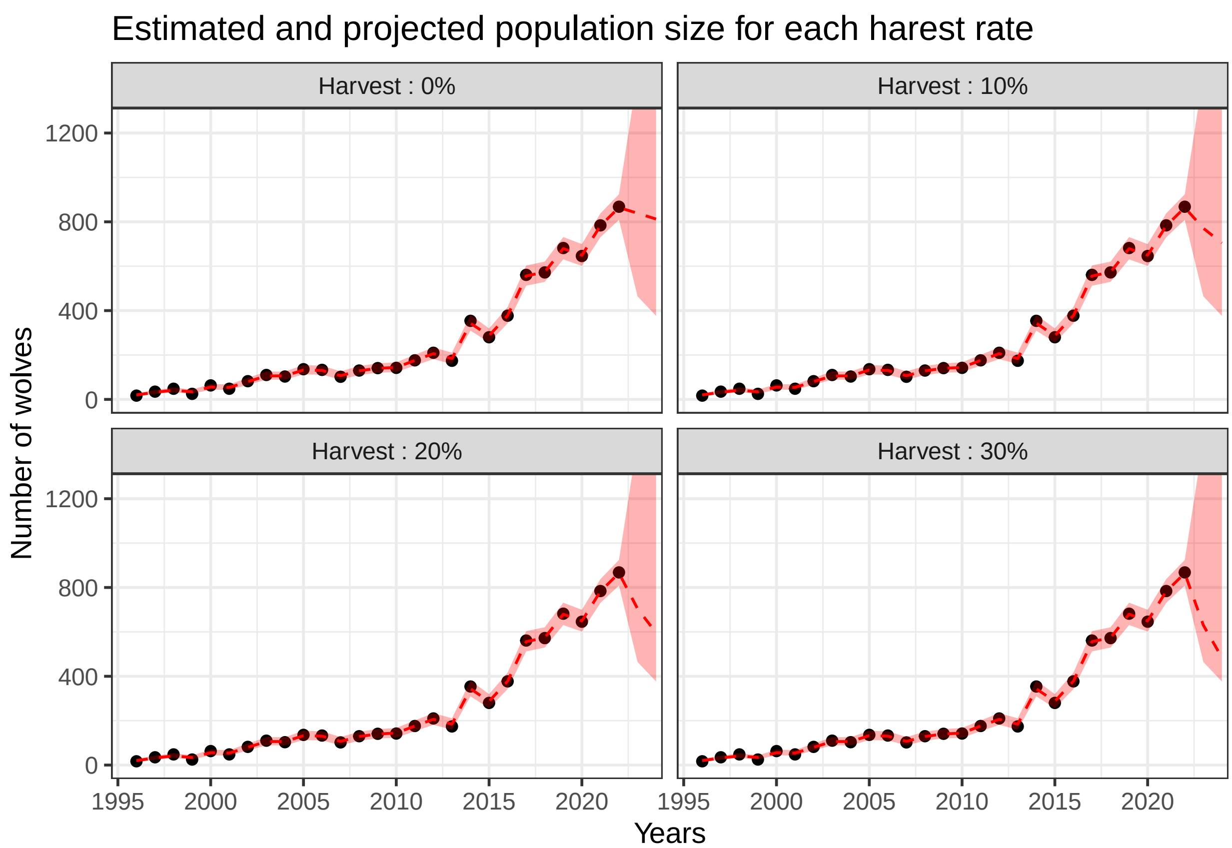
```
  mutate(years = parse_number(parameter3) + 1995)%>%
  arrange(years)
}

if(i==4){
  output4 = wolf_modellogist$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter4") %>%
  filter(str_detect(parameter4, "N")) %>%
  group_by(parameter4) %>%
  summarize(medianN4 = median(value),
            lq4 = quantile(value, probs = 2.5/100),
            hq4 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter4) + 1995)%>%
  arrange(years)
}
}
```

On affiche les estimations et projections pour chaque taux de prélevement :

```
output = output1 %>% left_join(output2) %>%
  left_join(output3) %>%
  left_join(output4) %>%
  pivot_longer(
    c(medianN1, medianN2, medianN3, medianN4),
    names_to = "medianN",
    values_to = "valuesM")
```

```
## Joining with `by = join_by(years)`
## Joining with `by = join_by(years)`
## Joining with `by = join_by(years)`
```

```
variable_names <- list(
  "medianN1" = "Harvest : 0%" ,
  "medianN2" = "Harvest : 10%",
  "medianN3" = "Harvest : 20%",
  "medianN4" = "Harvest : 30%")

variable_labeller <- function(variable, value) {
  return(variable_names[value])
}

  ggplot(output)+
  geom_point(aes(x = years, y = ObsY)) +
  coord_cartesian(xlim=c(1996,2023),ylim=c(0,1250))+
  aes(x = years, y = valuesM)+
  geom_line(colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq1, ymax = hq1), fill = "red", alpha = 0.3)+
  facet_wrap(~medianN,labeller = variable_labeller)+
  theme_bw()+
  labs(title = "Estimated and projected population size for each harest rate",
       x = "Years",
       y = "Number of wolves")
```



Comparaison DIC des deux modèles

Dans cette section on va comparer l'efficacité de chaque modèle selon le nombre de données, c'est-à-dire en fonction du temps passé.

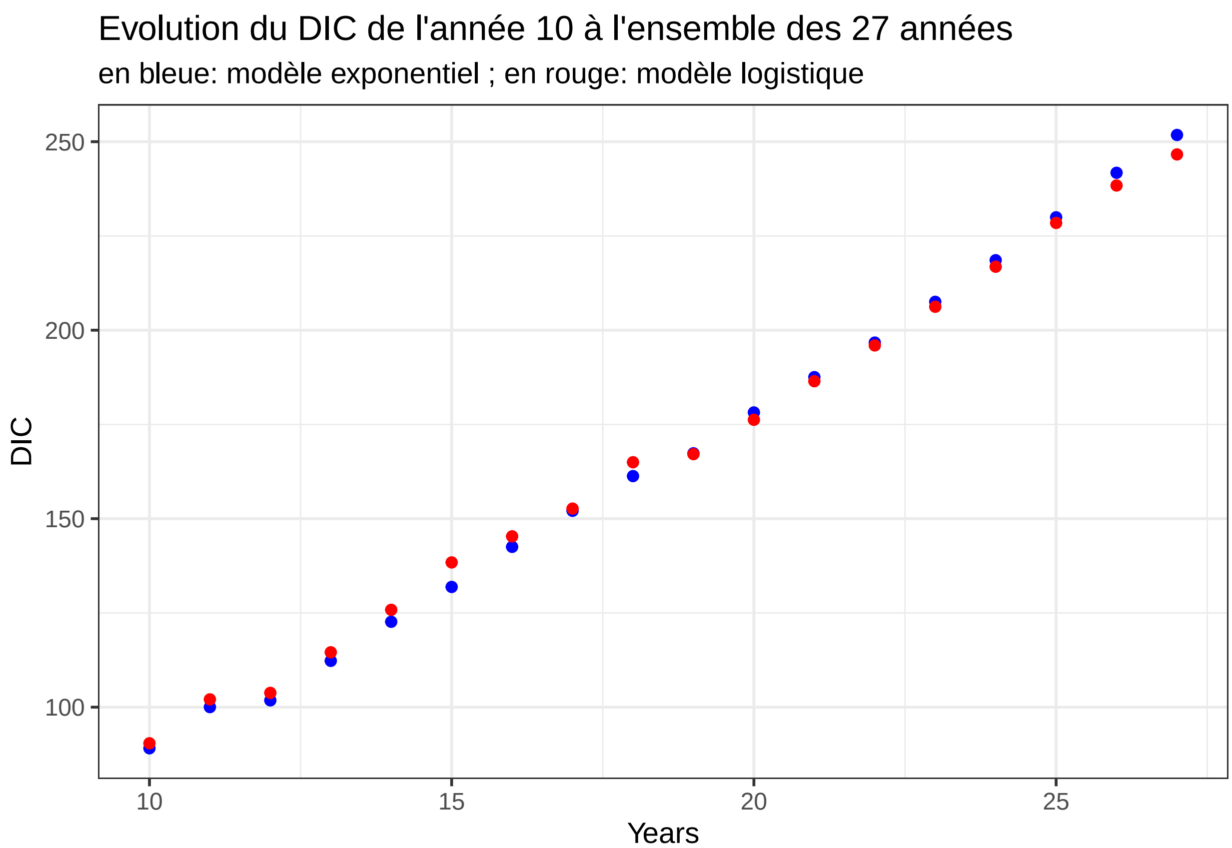
On stocke les résultats du DIC de chaque modèle de la 10ème année jusqu'à la fin.

```
DICexp=numeric(nrow(dat)-10)
DIClogist=numeric(nrow(dat)-10)

for (i in 10:nrow(dat)){
# Initialisation des données :
bugs.data = list(nyears = i,
                 y = dat$N[1:i],
                 h = dat$H[1:i])

# Modèle exponentiel
# Paramètres JAGS :
bugs.monitor = c("lambda", "sigmaProc", "N", "tauProc")
bugs.chains = 3
bugs.inits = function() {
  list()
}
#On lance la machine
wolf_modelexp = jags(data = bugs.data,
```

```r
                    inits = bugs.inits,
                    parameters.to.save = bugs.monitor,
                    model.file = modelexp,
                    n.chains = bugs.chains,
                    n.thin=10,
                    n.iter=50000,
                    n.burnin=20000)

# Enregistrement du DIC
DICexp[i-9]=wolf_modelexp$BUGSoutput$DIC

# Modèle logistique
# Paramètres JAGS
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)

# On lance la machine
wolf_modellogist = jags(data = bugs.data,
                    inits = bugs.inits,
                    parameters.to.save = bugs.monitor,
                    model.file = modellogist,
                    n.chains = bugs.chains,
                    n.thin=10,
                    n.iter=20000,
                    n.burnin=5000)

# Enregistrement du DIC
DIClogist[i-9]=wolf_modellogist$BUGSoutput$DIC
}
```

On affiche l'évolution des DIC des deux modèles au cours du temps.

```r
ggplot() +
  geom_point(aes(x = seq(10, 27), y = DICexp), colour = "blue") +
  geom_point(aes(x = seq(10, 27), y = DIClogist), colour = "red") +
  labs(
    title = "Evolution du DIC de l'année 10 à l'ensemble des 27 années",
    subtitle = "en bleue: modèle exponentiel ; en rouge: modèle logistique",
    x = "Years",
    y = "DIC") +
  theme_bw()
```



On constate que la différence d'efficacité entre les deux modèles n'est pas flagrante. Malgrès tout, le modèle exponentiel semble meilleur pour l'estimation des premières années, puis le modèle logistique est meilleur. Ce qui est logique avec la réalité biologique qui impose des limites d'espace et de ressources aux populations de loups. Celles-ci tendent donc à se stabiliser autour de la capacité de charge.

Simulation de données et prédiction

On va maintenant faire une simulation de données en fonction des paramètres obtenus par les estimations précédemment trouvées. On a une simulation suivant le modèle exponentiel, une autre suivant le modèle logistique, et une dernière suivant un mélange avec le modèle exponentiel sur les 15 premières années puis le modèle logistique sur la fin. Ces simulations permettent de voir si nos estimations précédentes peuvent bien reproduire un jeu de données similaire aux données observées. Nous pouvons ensuite faire une projection sur 20 ans afin de voir la dynamique proposée par chaque modèle.

Avec le modèle exponentiel

On initialise les paramètres pour la simulation des données

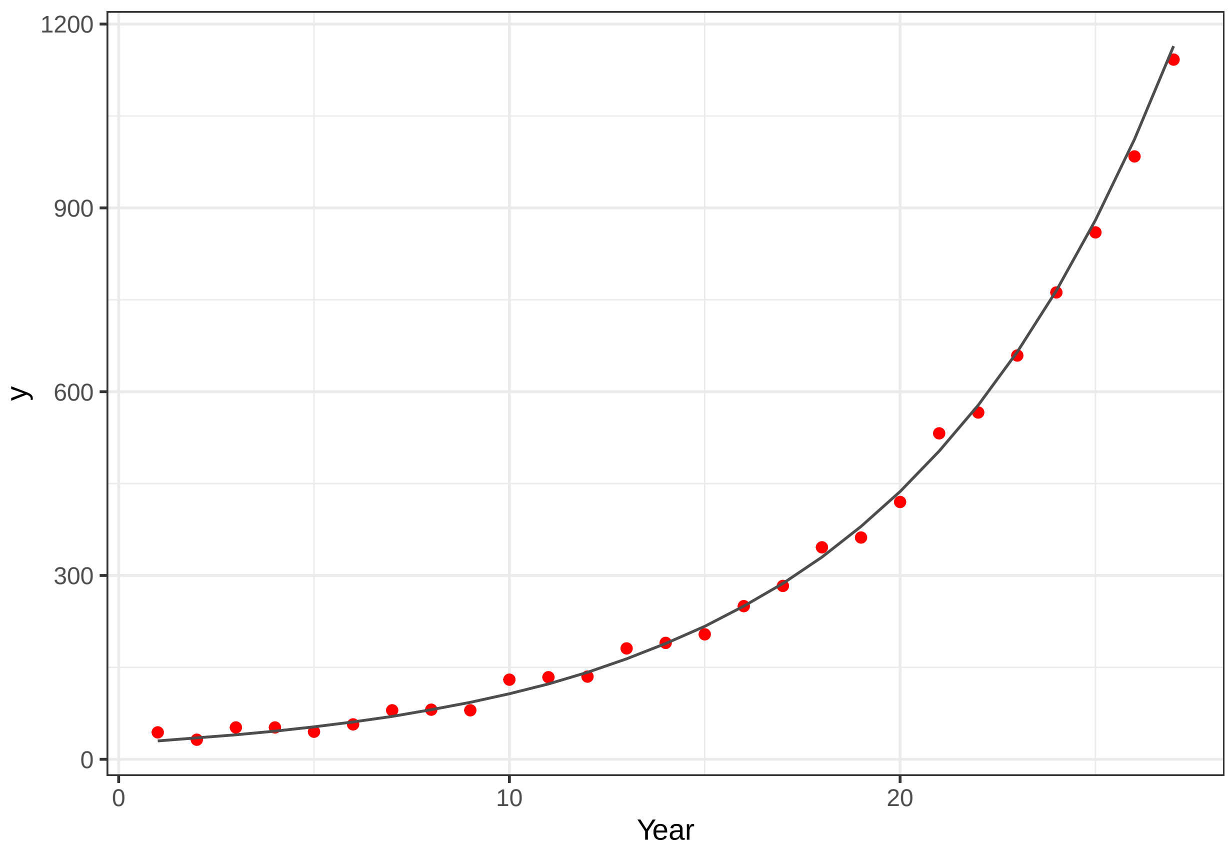
```
nyears = 27
N1 = 30
sigma = 0.15
lambda=1.15
```

On crée un data frame qui contiendra nos données.

```
ssm_sim1 <- data.frame(Year = 1:nyears,
                       y = numeric(nyears),
                       N = numeric(nyears))

ssm_sim1$N[1] = N1
```

On crée les données pas à pas en multipliant les effectifs par le taux de reproduction λ. On ajoute de la stochasticité avec $N_{t+1} \sim \text{Normale}(\lambda N_t, \sigma)$.

```
for (t in 1:(nyears-1)){
   ssm_sim1$N[t+1] <- round(rnorm(1,ssm_sim1$N[t] * lambda,sigma))
}

for (t in 1:nyears){
  ssm_sim1$y[t]=rpois(1,ssm_sim1$N[t])
}

ggplot(ssm_sim1, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "grey30") +
  theme_bw()
```



On va maintenant faire une estimation des données à l'aide du modèle exponentiel et projeter sur 20 ans.

```
modelexp = 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]
    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])
  }
}
```

Initialisation des données :

```
bugs.data = list(nyears = nrow(ssm_sim1)+20,
                 y = c(ssm_sim1$y,rep(NA,20)))
```

Paramètres JAGS :

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

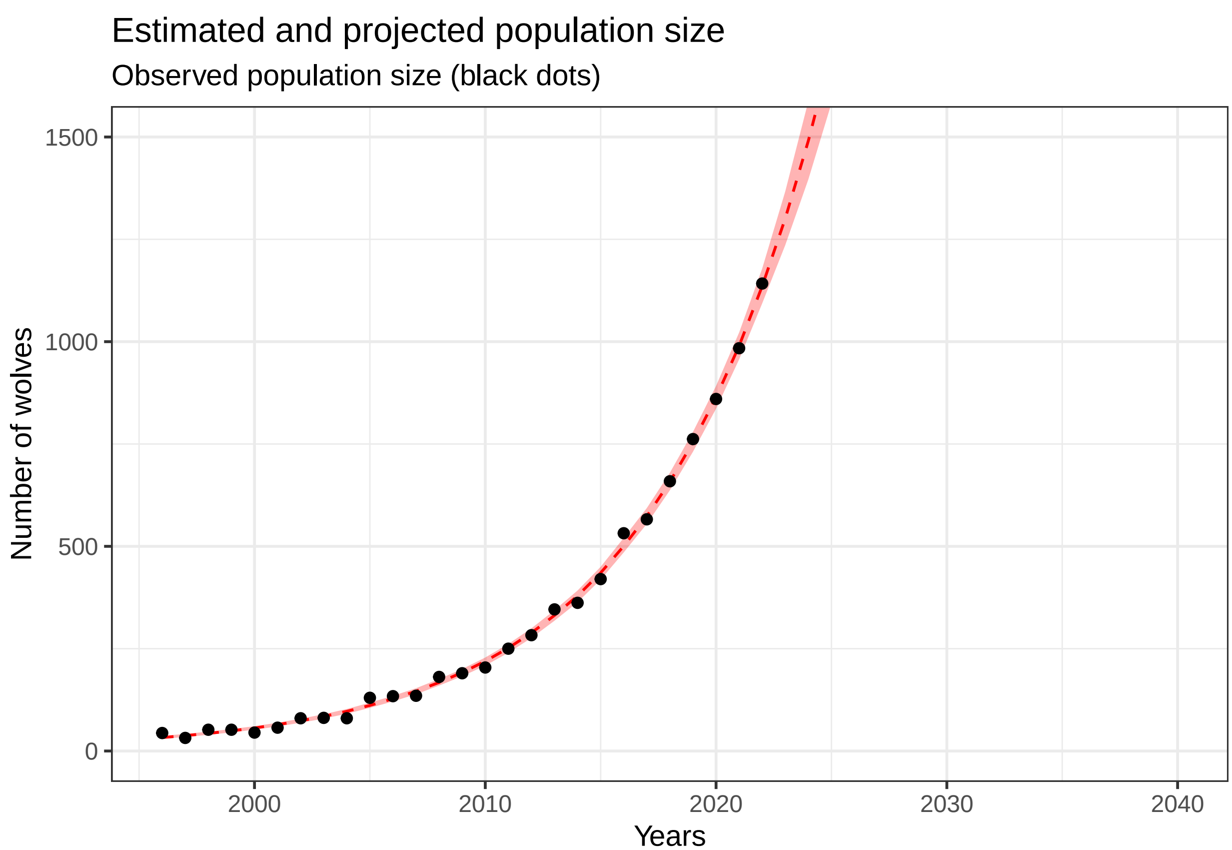
Lancement du modèle.

```
sim_modelexp = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modelexp,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=100000,
                   n.burnin=20000)
```

On affiche la dynamique de la population sur un graphique.

```
sim_modelexp$BUGSoutput$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),
          lq = quantile(value, probs = 2.5/100),
          hq = quantile(value, probs = 97.5/100))%>%
mutate(years = parse_number(parameter) + 1995)%>%
arrange(years)%>%
ggplot()+
geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
coord_cartesian(xlim=c(1996,2040),ylim=c(0,1500))+
theme_bw()+
labs(title = "Estimated and projected population size",
     subtitle = "Observed population size (black dots)",
     x = "Years",
     y = "Number of wolves")
```



Avec le modèle logistique

On initialise les paramètres pour la simulation des données

```
nyears = 27
N1 = 30
```

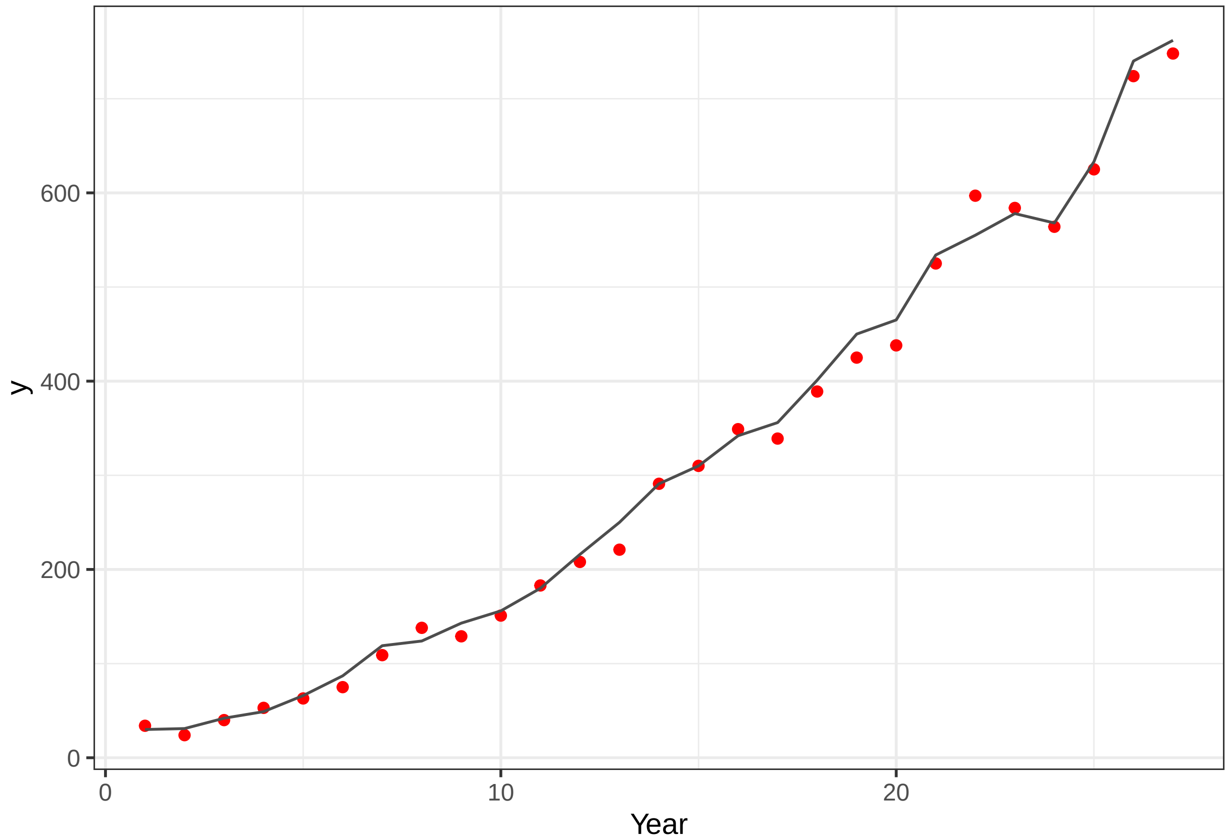
```
sigma = 0.15
K = 800
alpha = 0.2
```

On crée un data frame qui contiendra nos données

```
ssm_sim2 = data.frame(Year = 1:nyears,
                       y = numeric(nyears),
                       N = numeric(nyears))

ssm_sim2$N[1] = N1
```

On crée les données pas à pas avec le taux de reproduction $\lambda \sim \text{Normale}(\mu_\lambda, \sigma_\lambda)$ avec mu_λ et σ_λ définis plus tôt.

```
for (t in 1:(nyears-1)){
    Er = exp(alpha * (1 - ssm_sim2$N[t] / K)) * ssm_sim2$N[t]
    ssm_sim2$N[t+1] = rpois(1,Er)
}
for (t in 1:nyears){
  ssm_sim2$y[t]=rpois(1,ssm_sim2$N[t])
}


ggplot(ssm_sim2, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "grey30") +
  theme_bw()
```




```
modellogist = function() {
  # Priors
  sigmaProc ~ dunif (0, 10)
  tauProc = 1 / sigmaProc ^ 2
  alpha ~ dunif(0, 1.0986) #maximum exponential growth rate
  K ~ dunif(1, 1000)          #carrying capacity

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

  # Process model
  for (t in 2:(nyears)) {
    Er[t-1] = exp(alpha * (1 - N[t-1] / K))
    lambda[t-1] = N[t-1] * Er[t-1]
    N[t] ~ dpois(lambda[t-1])
  }
  # Observation model
  for (t in 1:nyears) {
    y[t] ~ dpois(N[t])
  }
}
```

Initialisation des données :

```
bugs.data = list(nyears = nrow(ssm_sim2)+20,
                 y = c(ssm_sim2$y,rep(NA,20)))
```

Paramètres JAGS :

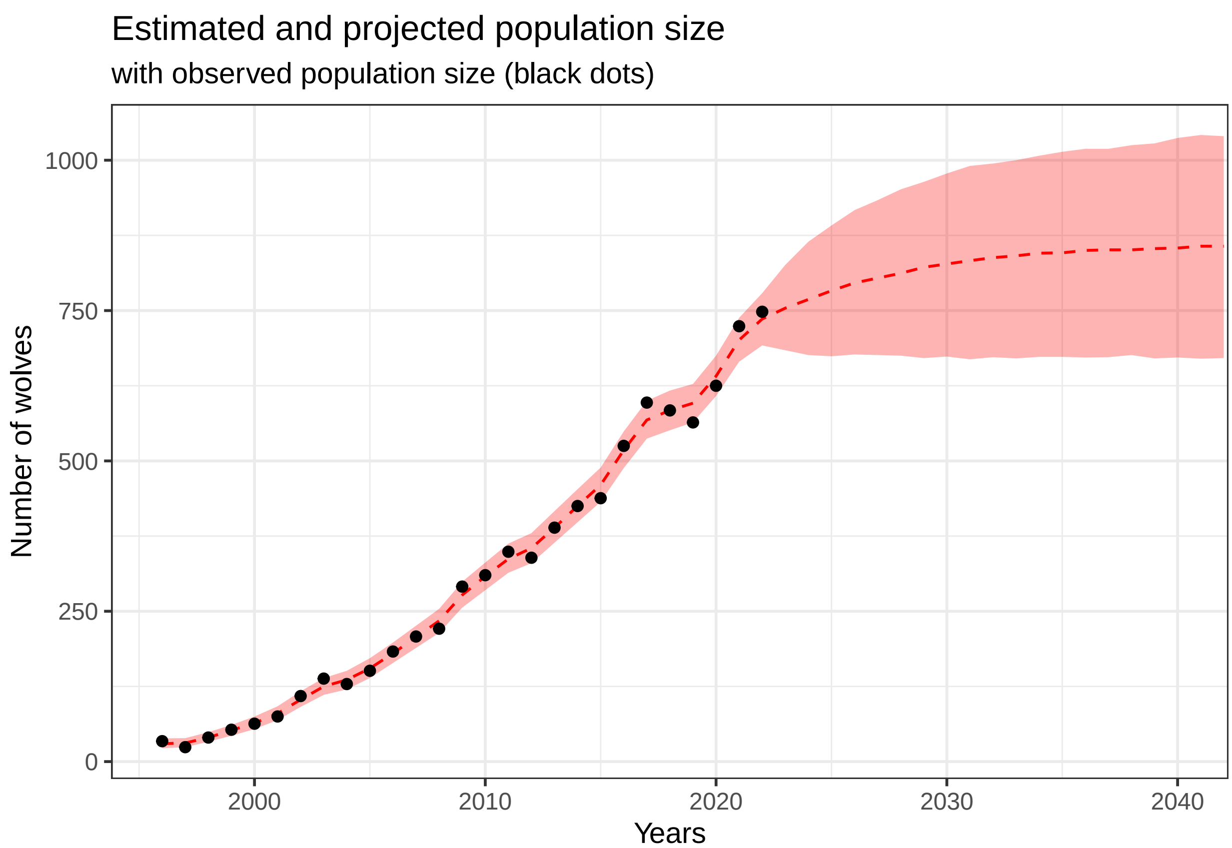
```
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "lambda", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)
```

Lancement du modèle.

```
sim_modellogist = jags(data = bugs.data,
                       inits = bugs.inits,
                       parameters.to.save = bugs.monitor,
                       model.file = modellogist,
                       n.chains = bugs.chains,
                       n.thin=10,
                       n.iter=20000,
                       n.burnin=5000)
```

On affiche la dynamique de la population sur un graphique.

```
sim_modellogist$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
            hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter) + 1995)%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  coord_cartesian(xlim=c(1996,2040))+
  theme_bw()+
  labs(title = "Estimated and projected population size",
       subtitle = "with observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```



En mélangeant les deux modèles

On initialise les paramètres pour la simulation des données

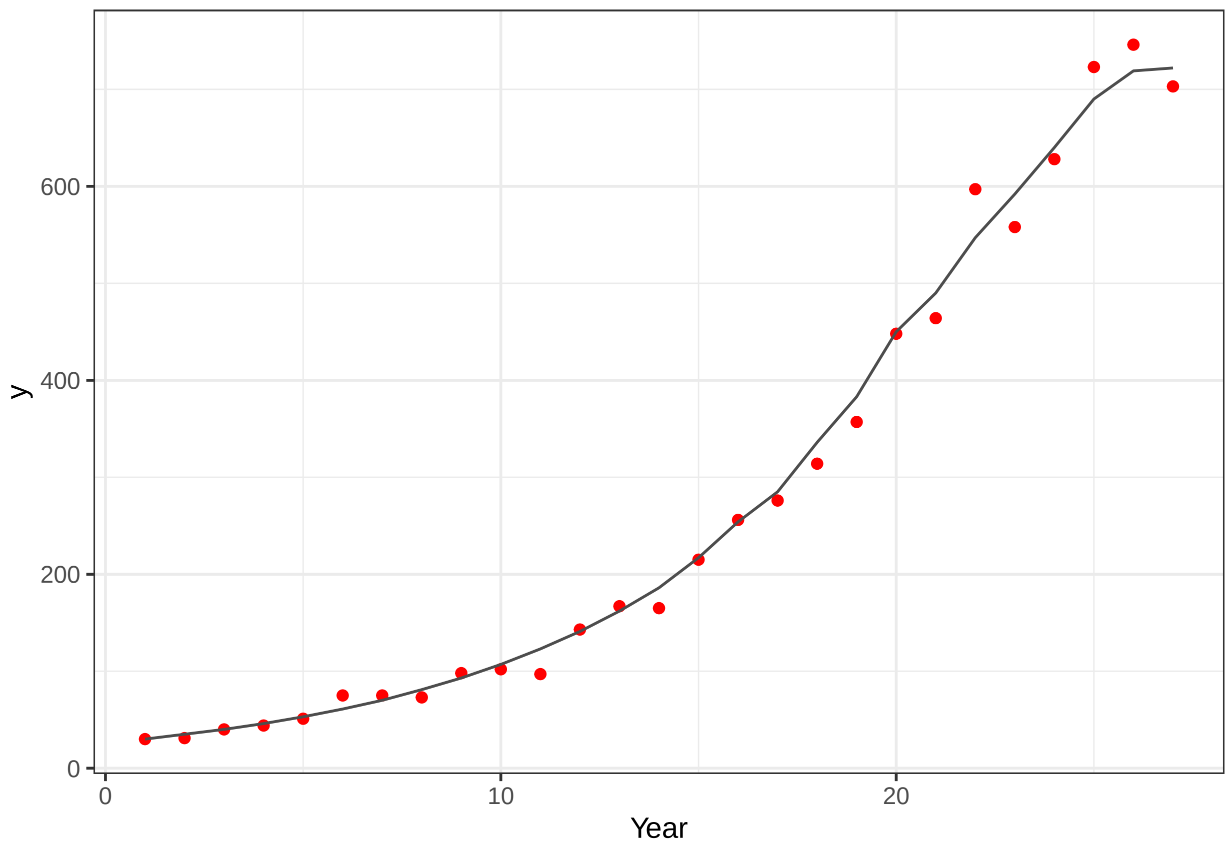
```
nyears = 27
N1 = 30
sigma = 0.15
lambda = 1.15
K = 800
alpha = 0.2
```

On crée un data frame qui contiendra nos données

```
ssm_sim3 = data.frame(Year = 1:nyears,
                      y = numeric(nyears),
                      N = numeric(nyears))

ssm_sim3$N[1] = N1
```

On crée les données pas à pas avec le taux de reproduction $\lambda \sim \text{Normale}(\mu_\lambda, \sigma_\lambda)$ avec mu_λ et σ_λ définis plus tôt.

```r
# Modèle exponentiel
for (t in 1:14){
   ssm_sim3$N[t+1] = round(rnorm(1,ssm_sim3$N[t] * lambda,sigma))
}
for (t in 1:15){
  ssm_sim3$y[t]=rpois(1,ssm_sim3$N[t])
}

# Modèle logistique
for (t in 15:nyears){
    Er = exp(alpha * (1 - ssm_sim3$N[t-1] / K)) * ssm_sim3$N[t-1]
    ssm_sim3$N[t] = rpois(1,Er)
}
for (t in 16:nyears){
  ssm_sim3$y[t]=rpois(1,ssm_sim3$N[t])
}

ggplot(ssm_sim3, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "grey30") +
  theme_bw()
```




```r
modellogist = function() {
  # Priors
  sigmaProc ~ dunif (0, 10)
```

```
  tauProc = 1 / sigmaProc ^ 2
  alpha ~ dunif(0, 1.0986) #maximum exponential growth rate
  K ~ dunif(1, 1000)         #carrying capacity

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

  # Process model
  for (t in 2:(nyears-20)) {
    Er[t-1] = exp(alpha * (1 - N[t-1] / K))
    lambda[t-1] = N[t-1] * Er[t-1]
    N[t] ~ dpois(lambda[t-1])
  }
  # Observation model
  for (t in 1:nyears) {
    y[t] ~ dpois(N[t])
  }
  # Projection
  for (t in (nyears-19):(nyears)) {
    Er[t-1] = exp(alpha * (1 - N[t-1] / K))
    lambda[t-1] = N[t-1] * Er[t-1]
    N[t] ~ dpois(lambda[t-1])
  }
}
```

Initialisation des données :

```
bugs.data = list(nyears = nrow(ssm_sim3)+20,
                 y = c(ssm_sim3$y,rep(NA,20)))
```

Paramètres JAGS :

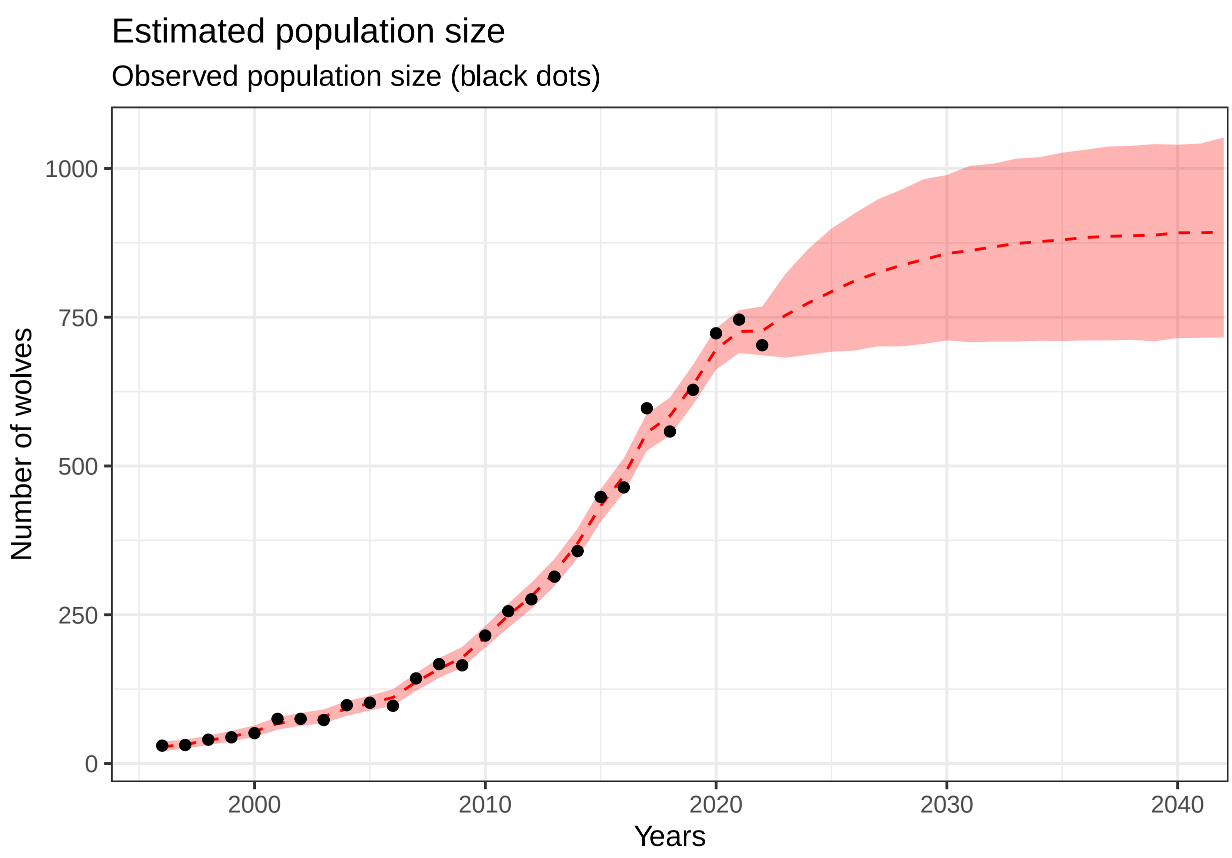
```
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "lambda", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)
```

Lancement du modèle.

```
sim_modellogist = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modellogist,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=20000,
                   n.burnin=5000)
```

On affiche la dynamique de la population sur un graphique.

```
sim_modellogist$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
            hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter) + 1995)%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  coord_cartesian(xlim=c(1996,2040))+
  theme_bw()+
  labs(title = "Estimated population size",
       subtitle = "Observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```



Simulation de gestion adaptative

Avec le modèle exponentiel

```
modelexp = 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)) {
    u[t-1] = N[t-1] * (lambda-dH)
    NProc[t] = log(max(1, u[t-1]))
    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+5)) {
    u[t-1] = N[t-1] * (lambda-dH)
    NProc[t] = log(max(1, u[t-1]))
    N[t] ~ dlnorm(NProc[t], tauProc)
  }
}
```

On crée un data frame qui contiendra nos premières données de simulation :

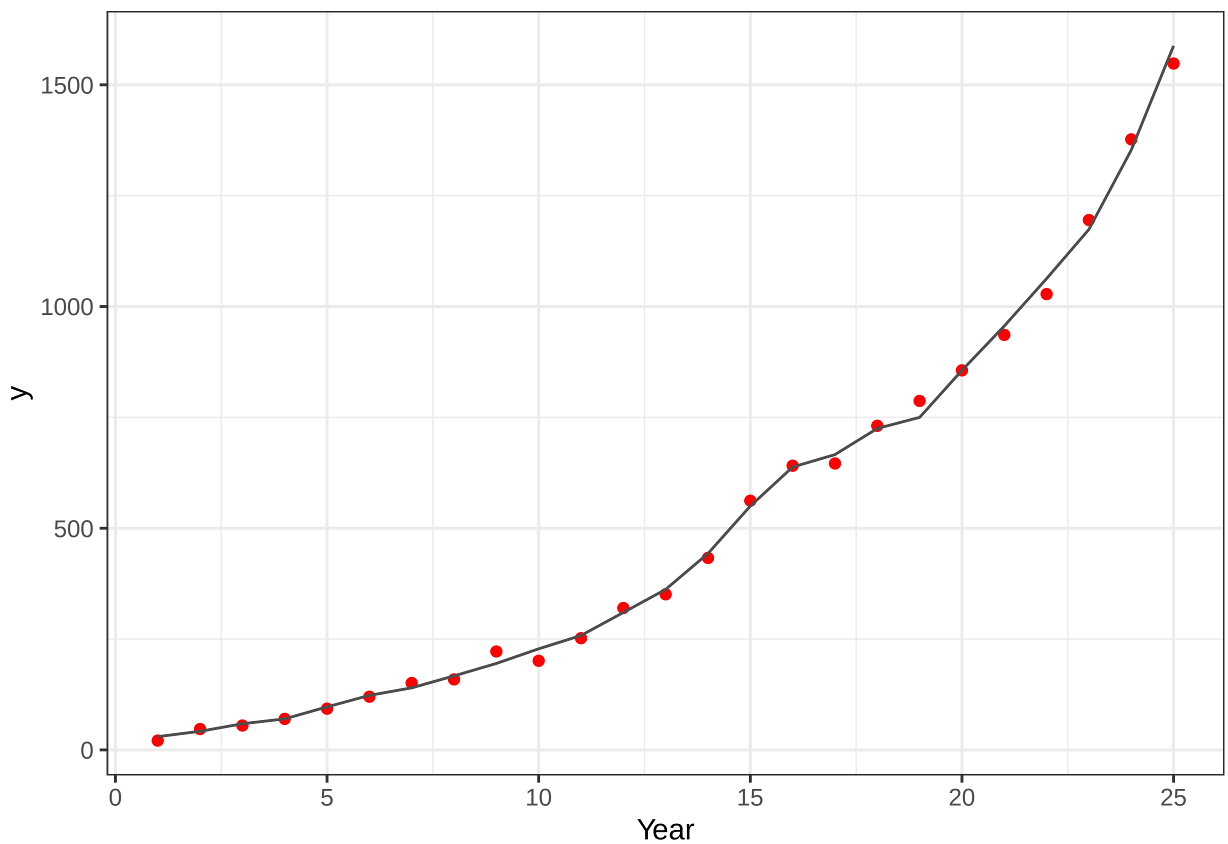
```
dH = c(0,10,20,30)/100
nyears = 25
N1 = 30

ssm_sim5 = data.frame(Year = 1:nyears,
                      y = numeric(nyears),
                      N = numeric(nyears))

ssm_sim5$N[1] = N1
```

```
H = 0
sigma = 0.15
ite = 0
tempH = c()
lambda = 1.2

for (nyears in seq(5, nyears, 5)) {
  # Boucle sur le nombre d'années
```

```r
print(nyears)
ite = ite + 1
tempH[ite] = H

if (nyears == 5) {
  # Simulation des 5 premières années
  for (t in 1:(nyears - 1)) {
    u = ssm_sim5$N[t] * (lambda - H)
    ssm_sim5$N[t + 1] = rpois(1, u)
  }
}

if (nyears > 5) {
  # Simulation des années suivantes, 5 par 5
  for (t in (nyears - 5):(nyears - 1)) {
    u = ssm_sim5$N[t] * (lambda - H)
    ssm_sim5$N[t + 1] = rpois(1, u)
  }
}

for (t in 1:nyears) {
  # Simulation des données observées
  ssm_sim5$y[t] = rpois(1, ssm_sim5$N[t])
}

# Initialisation des données
bugs.data = list(nyears = nyears,
                 y = c(ssm_sim5$y[1:nyears], rep(NA, 5)),
                 dH = H)
# Paramètres JAGS
bugs.monitor = c("sigmaProc", "tauProc", "lambda", "N")
bugs.chains = 3
bugs.inits = function() {
  list()
}

# Lancement du modèle
wolf_modelexp = jags(
  data = bugs.data,
  inits = bugs.inits,
  parameters.to.save = bugs.monitor,
  model.file = modelexp,
  n.chains = bugs.chains,
  n.thin = 10,
  n.iter = 100000,
  n.burnin = 20000
)

#print(wolf_modelexp, intervals = c(2.5 / 100, 50 / 100, 97.5 / 100))

# Taux de reproduction estimé
lambda = wolf_modelexp$BUGSoutput$median$lambda # lambda estimé sur une période les données observées
print(lambda)
```

```
  if (lambda<1.2){H=0}
  if(lambda>=1.2 & lambda<1.3){H=0.1}
  if(lambda>=1.3 & lambda<1.4){H=0.2}
  if(lambda>1.4){H=0.3}

  print(H)
}
```

```
ggplot(ssm_sim5, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "grey30") +
  theme_bw()
```

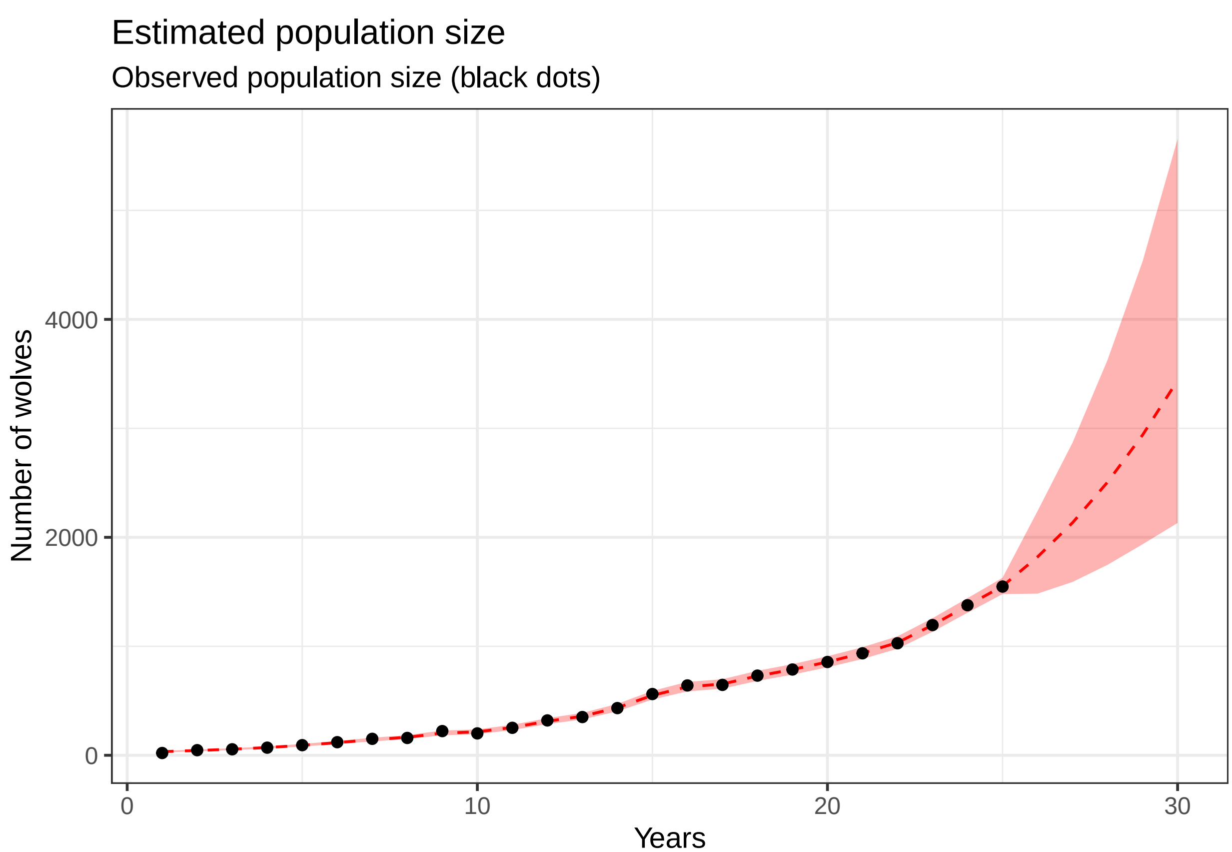



```
tempH
```

```
## [1] 0.0 0.0 0.1 0.2 0.2
```

```
wolf_modelexp$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
```

```
             hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter))%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1:unique(nyears+5), y = c(ssm_sim5$y,rep(NA,5)))) +
#  coord_cartesian(ylim=c(0,1500))+
  theme_bw()+
  labs(title = "Estimated population size",
       subtitle = "Observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```



Avec le modèle logisitique

```
modellogist = function() {
  # Priors
  sigmaProc ~ dunif (0, 5)
  tauProc = 1 / sigmaProc ^ 2
  alpha ~ dunif(0, 1.0986) #maximum exponential growth rate
  K ~ dunif(1, 1000)          #carrying capacity
```

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

  # Process model
  for (t in 2:(nyears+5)) {
    u[t-1] = N[t-1] * (1-dH)
    Er[t] = exp(alpha * (1 - u[t-1] / K)) # per capita growth rate is density dependent - Ricker model
    lambda[t] = u[t-1] * Er[t]
    NProc[t] = log(max(1, lambda[t]))
    N[t] ~ dlnorm(NProc[t], tauProc)
  }
  # Observation model
  for (t in 1:(nyears)) {
    y[t] ~ dpois(N[t])
  }
}
```

On crée un data frame qui contiendra nos premières données de simulation :

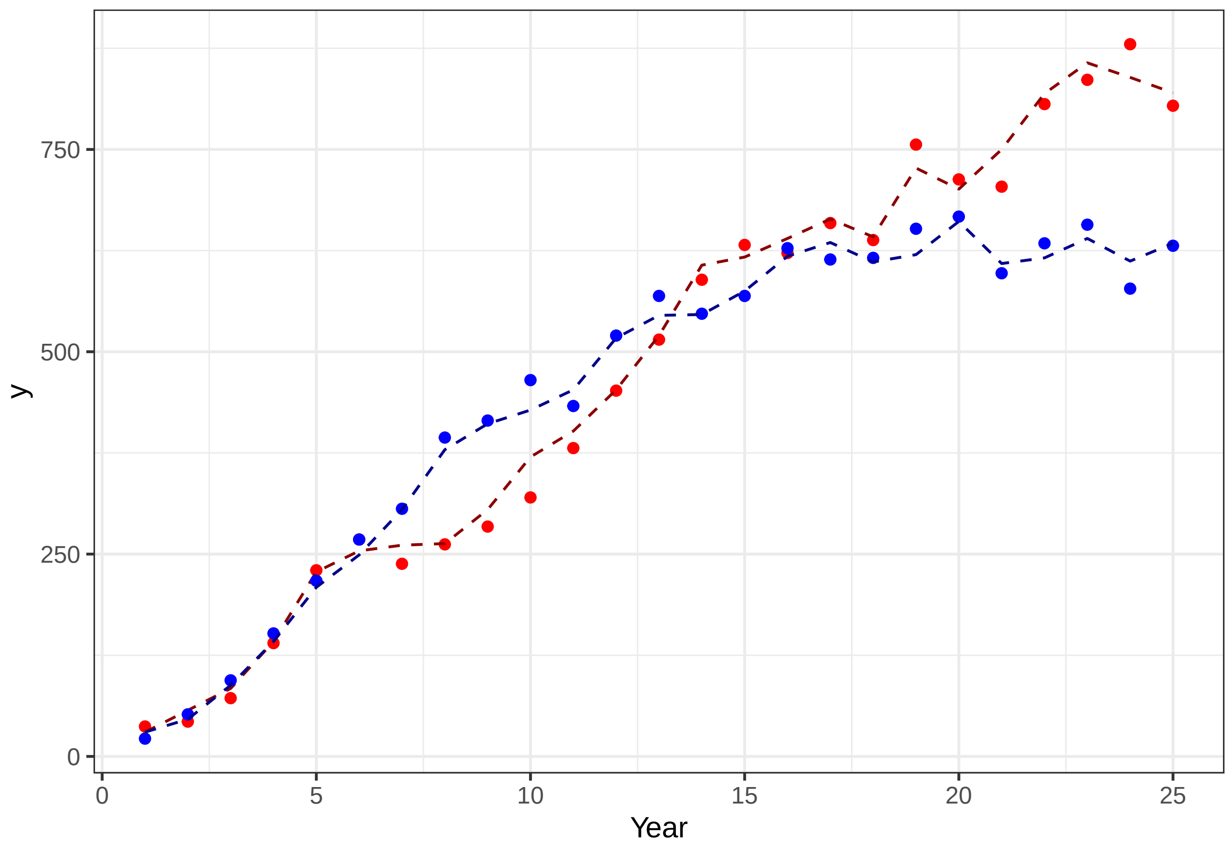
```
nyears = 25
N1 = 30

ssm_sim4 = data.frame(Year = 1:nyears,
                       y = numeric(nyears),
                       N = numeric(nyears),
                      ybis = numeric(nyears),
                      Nbis = numeric(nyears))

ssm_sim4$N[1] = N1
ssm_sim4$Nbis[1] = N1
```

```
# Paramètres initiaux
pas = 1
H = 0
sigma = 0.15
K = 800
alpha = 0.5
ite = 0
tempH = c()
NAMharvest = 0.15

# Lancement du modèle
for (nyears in seq(2,nyears,pas)) { # Boucle sur le nombre de tranches d'années parcourues
  ite = ite+1
  tempH[ite] = H # Enregistre les taux de prélevement pour chaque année

  # Simulation des effectifs
  if (nyears <= 5) { # Initialisation des effectifs, sans prélevement sur les 5 premières années
    for (t in 1:(nyears - 1)) {
      u = ssm_sim4$N[t]

      Er = exp(alpha * (1 - u/ K)) * u

      ssm_sim4$N[t+1] = rpois(1,Er)
```

```
    ssm_sim4$Nbis[t+1] = rpois(1,Er)
  }
}

if(nyears > 5){ # Suite de la simulation des effectifs entre les années 6 et 25
  for (t in (nyears - pas):(nyears - 1)) {
    u = ssm_sim4$N[t]*(1-H)             # Taux de prélevement adaptatif
    v = ssm_sim4$Nbis[t]*(1-NAMharvest) # Taux de prélevement constant

    Er = exp(alpha * (1 - c(u,v)/ K)) * c(u,v)

    ssm_sim4$N[t+1] = rpois(1,Er[1])
    ssm_sim4$Nbis[t+1] = rpois(1,Er[2])
    }
  }
# Simulation des effecitfs observés
for (t in 1:nyears){
  ssm_sim4$y[t]=rpois(1,ssm_sim4$N[t])
  ssm_sim4$ybis[t]=rpois(1,ssm_sim4$Nbis[t])
}

# Début de l'estimation par approche bayésienne
  # Initialisation des données
  bugs.data = list(nyears = nyears,
                   y = c(ssm_sim4$y[1:nyears],rep(NA,5)),
                   dH = H)

  # Paramètres JAGS
  bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "N")
  bugs.chains = 3
  init1 = list(alpha = .5, sigmaProc = .25)
  init2 = list(alpha = .1, sigmaProc = .05)
  init3 = list(alpha = 1, sigmaProc = .45)
  bugs.inits = list(init1, init2, init3)

  # Lancement du modèle

  wolf_modellogist = jags(
    data = bugs.data,
    inits = bugs.inits,
    parameters.to.save = bugs.monitor,
    model.file = modellogist,
    n.chains = bugs.chains,
    n.thin = 10,
    n.iter = 20000,
    n.burnin = 5000
  )
  output1 = wolf_modellogist$BUGSoutput$sims.matrix

  # Calcul du taux de reproduction estimé
  Nest = wolf_modellogist$BUGSoutput$median$N
  l = length(Nest)
  lamb = c()
```

```r
    for (t in 1:(l-5)) {
      lamb[t] = Nest[t+1] / Nest[t]}
    lambda = mean(lamb)
    print(lambda)

    # Conditions de modification du taux de prélevement
    if (lambda < 1.2) {H = 0}
    if (lambda >= 1.2 & lambda < 1.3) {H = 0.1}
    if (lambda >= 1.3 & lambda < 1.4) {H = 0.2}
    if (lambda > 1.4) {H = 0.3}

  print(H)
}
```

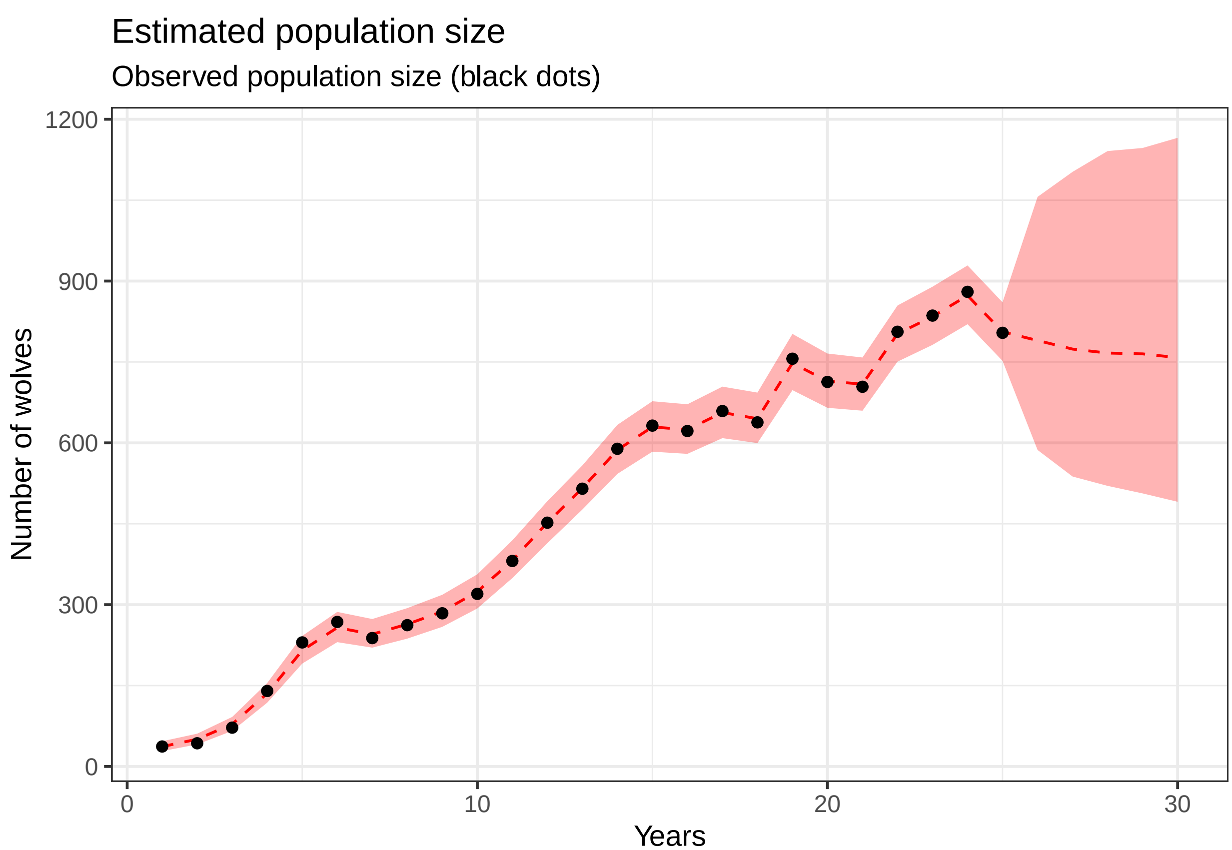
```r
ggplot(ssm_sim4, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "red4", lty = "dashed") +
  geom_point(aes(y = ybis), colour = "blue") +
  geom_line(aes(y = Nbis), colour = "blue4", lty = "dashed") +
  theme_bw()
```




```r
tempH
```

```
##  [1] 0.0 0.2 0.2 0.3 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.0 0.0
## [20] 0.0 0.0 0.0 0.0 0.0
```

```r
wolf_modellogist$BUGSoutput$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),
            lq = quantile(value, probs = 2.5/100),
            hq = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter))%>%
  arrange(years)%>%
  ggplot()+
  geom_line(aes(x = years, y = medianN), colour = "red", lty = "dashed")+
  geom_ribbon(aes(x = years, ymin = lq, ymax = hq), fill = "red", alpha = 0.3)+
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1:unique(nyears+5), y = c(ssm_sim4$y,rep(NA,5))))
  theme_bw()+
  labs(title = "Estimated population size",
       subtitle = "Observed population size (black dots)",
       x = "Years",
       y = "Number of wolves")
```




```r
tempH
```

```
##  [1] 0.0 0.2 0.2 0.3 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.0 0.0
## [20] 0.0 0.0 0.0 0.0 0.0
```

Comparaison avec une gestion non-adaptative

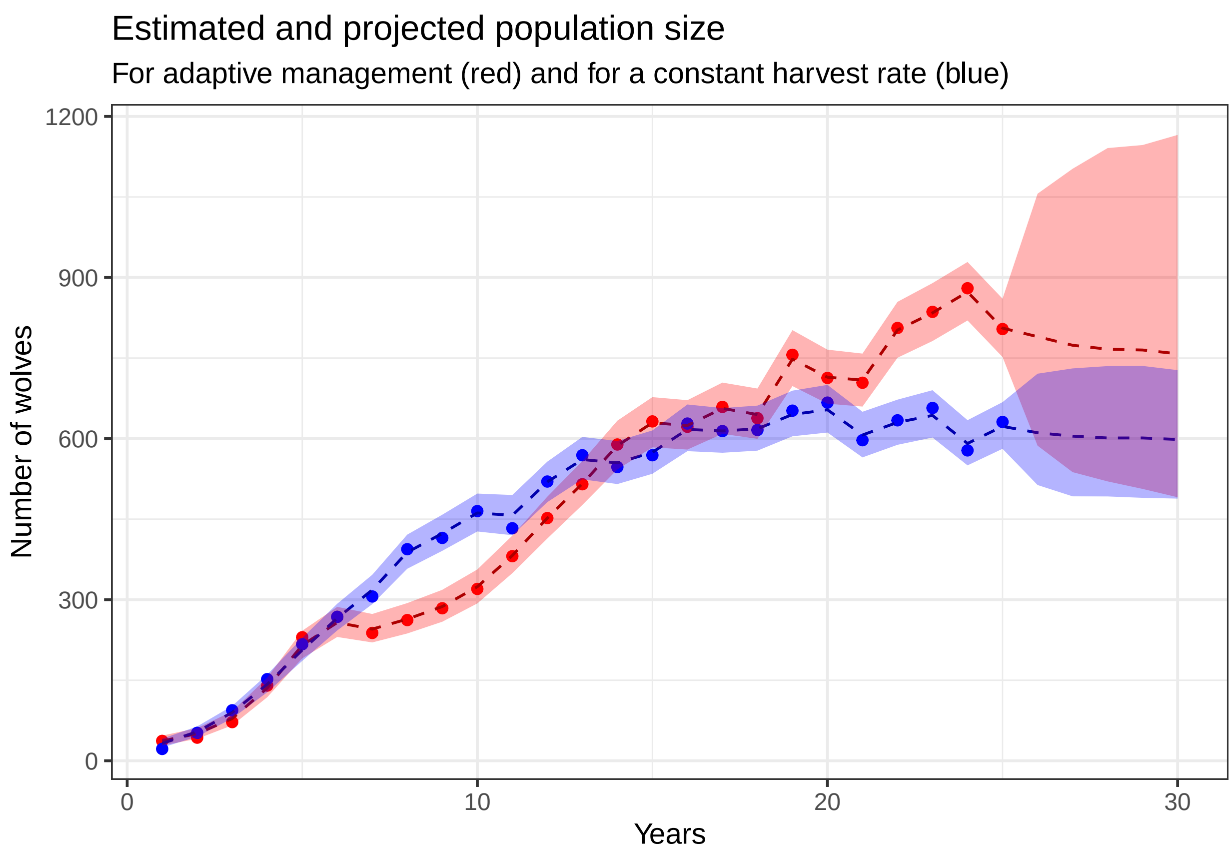
On prend le même jeu de données simulé pécédemment et on applique un taux de prélevement constant chaque année (15% pour coller avec le taux de prélevement observé sur les loups en France).

```
# Initialisation des données
bugs.data = list(nyears = nyears,
                 y = c(ssm_sim4$ybis[1:nyears], rep(NA, 5)),
                 dH = NAMharvest)

wolf_modellogist = jags(
  data = bugs.data,
  inits = bugs.inits,
  parameters.to.save = bugs.monitor,
  model.file = modellogist,
  n.chains = bugs.chains,
  n.thin = 10,
  n.iter = 20000,
  n.burnin = 5000
)

output2 = wolf_modellogist$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),
               values_to = "value",
               names_to = "parameter2") %>%
  filter(str_detect(parameter2, "N")) %>%
  group_by(parameter2) %>%
  summarize(
    medianN2 = median(value),
    lq2 = quantile(value, probs = 2.5 / 100),
    hq2 = quantile(value, probs = 97.5 / 100)
  ) %>%
  mutate(years = parse_number(parameter2)) %>%
  arrange(years)
```

```
output1 = output1 %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),
               values_to = "value",
               names_to = "parameter1") %>%
  filter(str_detect(parameter1, "N")) %>%
  group_by(parameter1) %>%
  summarize(
    medianN1 = median(value),
    lq1 = quantile(value, probs = 2.5 / 100),
    hq1 = quantile(value, probs = 97.5 / 100)
  ) %>%
  mutate(years = parse_number(parameter1)) %>%
  arrange(years)


output = output1 %>% left_join(output2)
```

```
## Joining with ‘by = join_by(years)‘
```

```r
ggplot(output) +
geom_point(data = ssm_sim4, aes(x = seq(1, 25), y = y), colour = "red") +
geom_point(data = ssm_sim4, aes(x = seq(1, 25), y = ybis), colour = "blue")+
geom_line(aes(x = years, y = medianN1), colour = "red4", lty = "dashed")+
geom_line(aes(x = years, y = medianN2), colour = "blue4", lty = "dashed")+
geom_ribbon(aes(x = years, ymin = lq1, ymax = hq1), fill = "red", alpha = 0.3)+
geom_ribbon(aes(x = years, ymin = lq2, ymax = hq2), fill = "blue", alpha = 0.3)+
# facet_wrap(~medianN,labeller = variable_labeller)+
theme_bw()+
labs(title = "Estimated and projected population size",
     subtitle = "For adaptive management (red) and for a constant harvest rate (blue)",
     x = "Years",
     y = "Number of wolves")
```




```r
# Ranger les taux de croissances de chaque méthode
lambda1=c()
lambda2=c()

for (t in 1:(nyears-1)){
  lambda1[t]=output1$medianN1[t+1]/output1$medianN1[t]
  lambda2[t]=output2$medianN2[t+1]/output2$medianN2[t]
}
```

```r
# Compter le nombre de fois que le taux de croissance n'est pas dans l'objectif [1,1.2]
fail1=0
fail2=0
for (t in 1:length(lambda1)){
  if (between(lambda1[t],1,1.2)==FALSE){fail1=fail1+1}
  if (between(lambda2[t],1,1.2)==FALSE){fail2=fail2+1}
}

fail1
```

```
## [1] 10
```

```r
fail2
```

```
## [1] 11
```