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 error
```

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
```

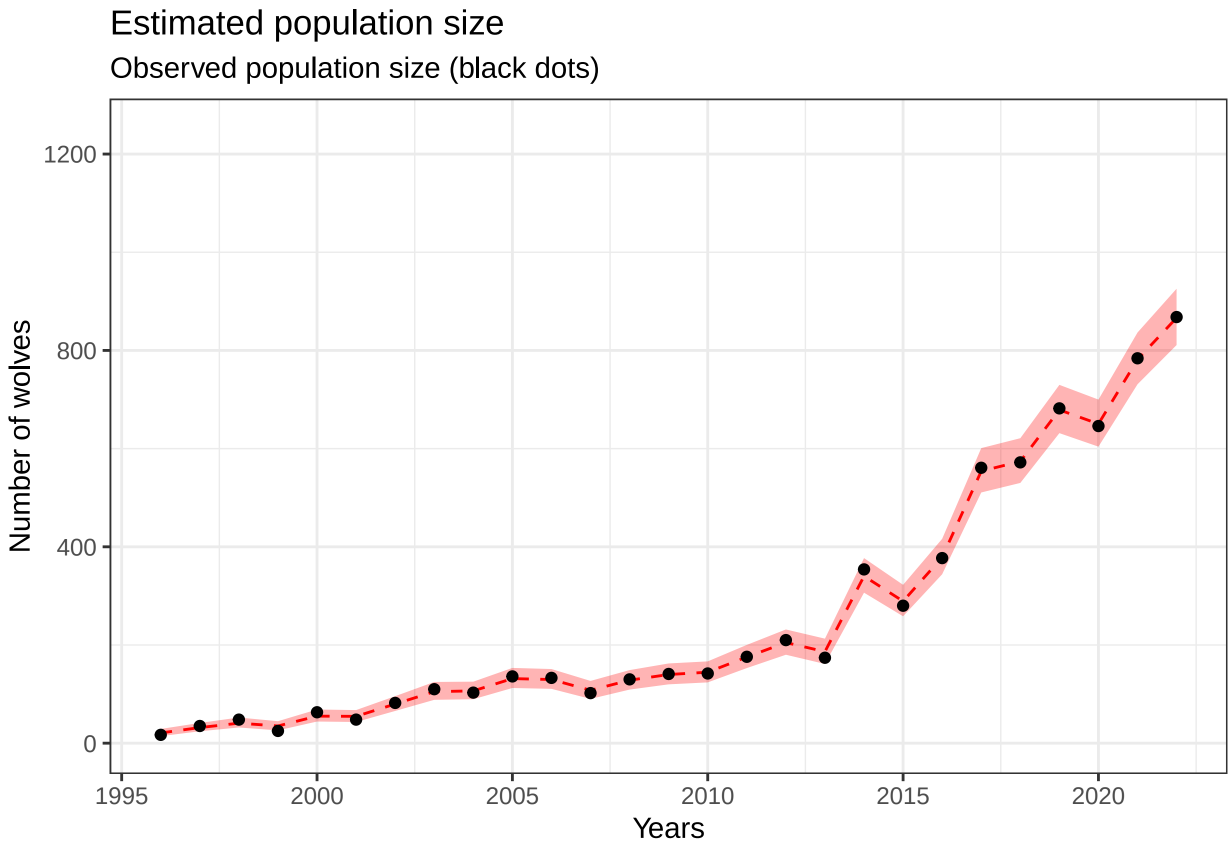
On affiche les estimations obtenus.

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

```
## Inference for Bugs model at "/tmp/RtmpKfsdGJ/model378227ad59c3.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]       21.047   3.769  14.307  20.853  29.035 1.001 12000
## N[2]       32.021   4.360  24.265  31.782  41.226 1.001 15000
## N[3]        41.142   5.237  31.946  40.765  52.322 1.001  7900
## N[4]        35.156   4.747  26.088  35.026  44.820 1.001 15000
## N[5]        55.413   6.240  44.056  55.087  68.611 1.001  6300
## N[6]        54.872   6.180  43.165  54.669  67.504 1.001  8500
## N[7]        80.067   7.684  65.642  79.802  95.805 1.001 12000
## N[8]       105.404   9.178  88.309 104.937 124.721 1.001 15000
## N[9]       106.778   9.185  89.537 106.509 125.347 1.001 15000
## N[10]      131.927  10.405 112.373 131.567 153.351 1.001 15000
## N[11]      130.046  10.235 110.682 129.694 151.085 1.001 15000
## N[12]      107.998   9.283  90.177 107.805 126.822 1.001 12000
## N[13]      128.510  10.108 109.151 128.291 148.919 1.001 15000
## N[14]      140.126  10.797 119.843 139.865 162.405 1.001 15000
## N[15]      144.684  10.912 123.805 144.533 166.687 1.001  5400
## N[16]      175.786  12.153 152.950 175.484 200.231 1.001 15000
## N[17]      205.254  13.218 180.145 204.899 231.667 1.001 12000
## N[18]      186.585  13.197 161.338 186.445 212.951 1.001 15000
## N[19]      340.641  18.008 306.528 340.245 377.030 1.001 15000
## N[20]      289.510  16.427 258.436 289.125 322.585 1.001 11000
## N[21]      378.923  18.471 344.040 378.636 416.304 1.001 15000
## N[22]      554.465  23.123 510.669 554.105 601.022 1.001  3800
## N[23]      574.546  23.224 530.101 574.019 621.266 1.001 15000
## N[24]      679.363  25.230 631.499 678.863 729.888 1.001  6200
## N[25]      650.861  24.511 603.989 650.373 699.893 1.001 13000
## N[26]      782.118  27.203 731.013 781.559 836.506 1.001 15000
## N[27]      866.977  29.283 810.911 866.459 925.737 1.001 15000
## lambda       1.207   0.062   1.090   1.204   1.336 1.001 15000
## sigmaProc    0.250   0.053   0.164   0.243   0.371 1.001 15000
## tauProc     18.234   7.788   7.250  16.874  37.227 1.001 15000
## deviance   218.754   8.077 204.702 218.179 236.192 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 = 32.6 and DIC = 251.4
## DIC is an estimate of expected predictive error (lower deviance is better).
```

On affiche la dynamique de la population sur un graphique.

```
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") %>%
```

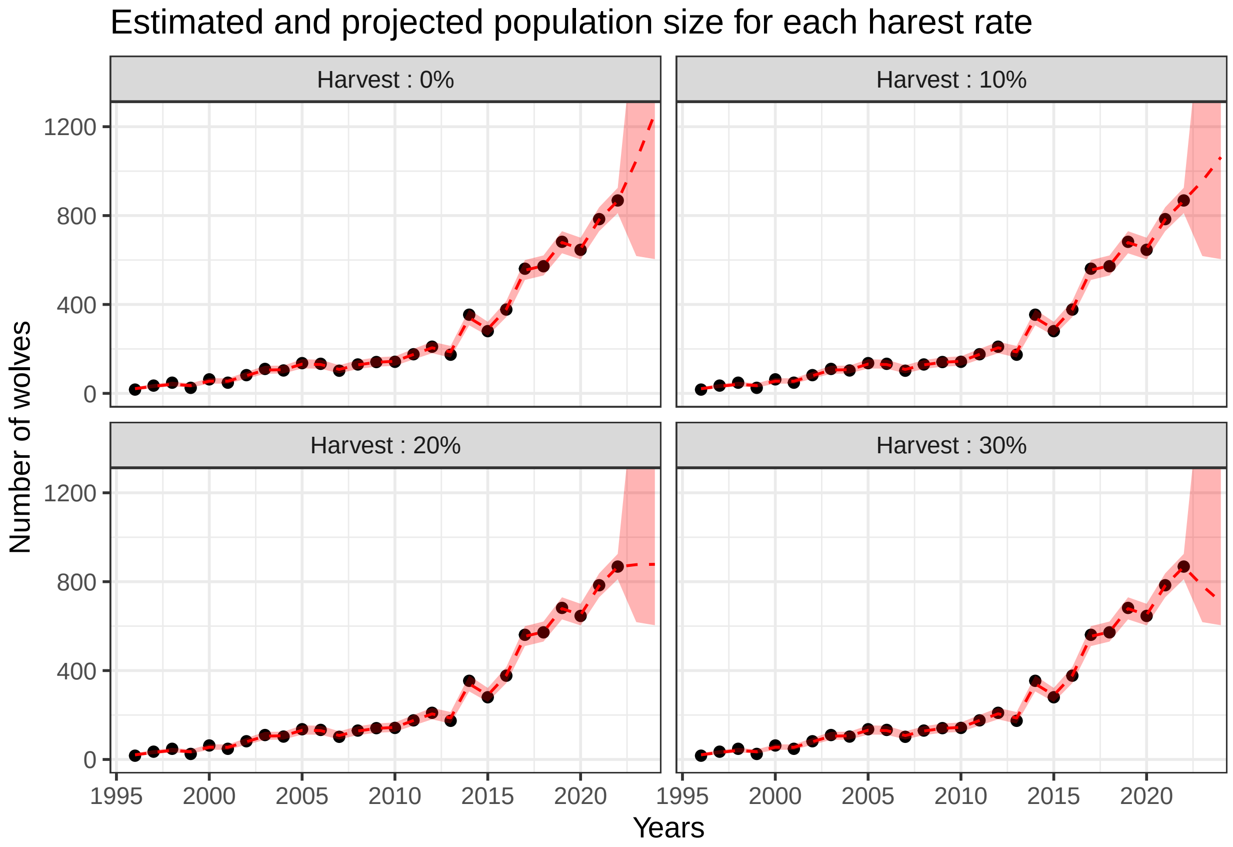
```r
  filter(str_detect(parameter1, "N")) %>%
  group_by(parameter1) %>%
  summarize(medianN1 = median(value),
            lq1 = quantile(value, probs = 2.5/100),
            hq1 = quantile(value, probs = 97.5/100))%>%
  mutate(years = parse_number(parameter1) + 1995)%>%
  arrange(years)%>%
  mutate(ObsY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

```
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 :

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

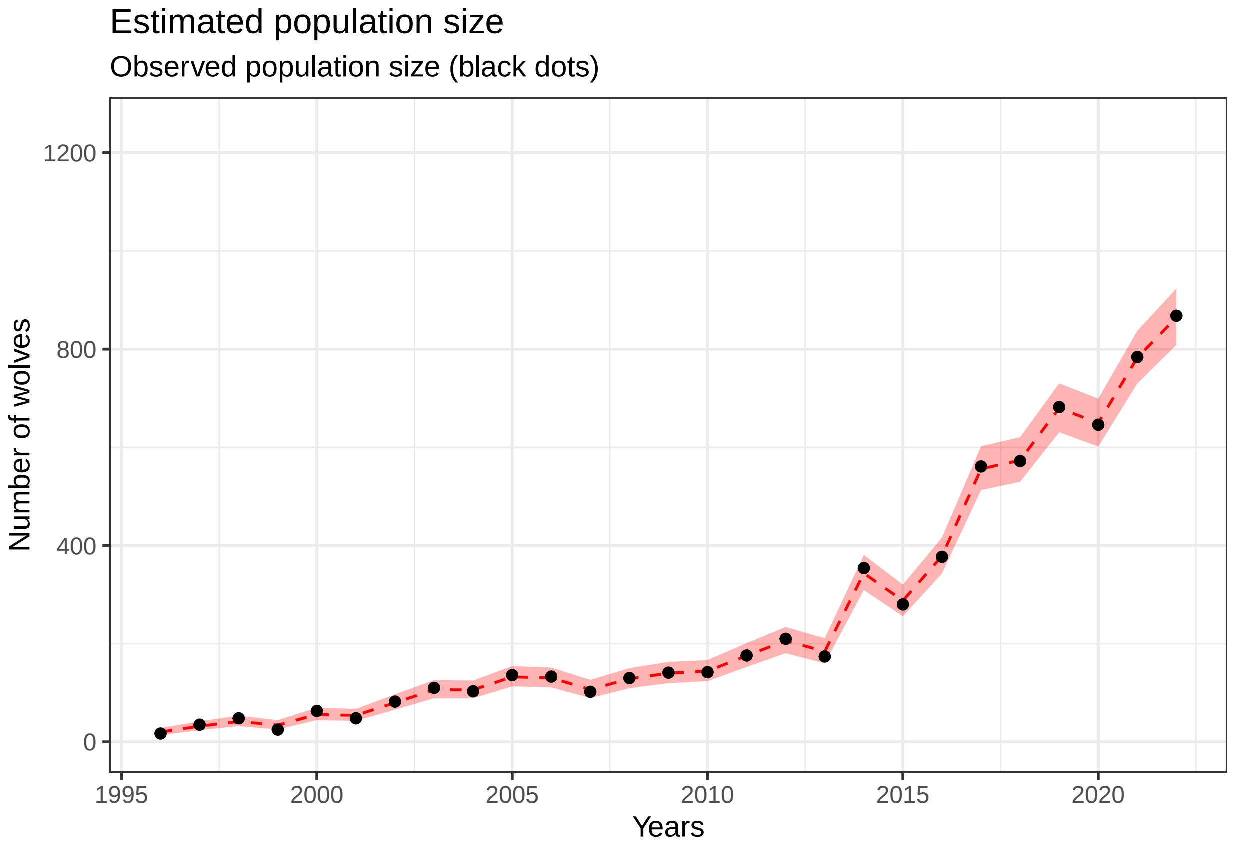
On affiche les estimations obtenus.

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

```
## Inference for Bugs model at "/tmp/RtmpKfsdGJ/model37824432b924.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         780.014 155.635 439.296 804.263 991.923 1.001  6300
## N[1]       20.324   3.756  13.637  20.111  28.373 1.001 12000
## N[2]       32.178   4.561  23.823  31.966  41.904 1.001 11000
## N[3]       41.767   5.406  32.057  41.440  53.372 1.001 11000
## N[4]       34.254   4.870  25.190  34.082  44.344 1.001 12000
## N[5]       55.983   6.489  44.329  55.704  69.730 1.001 11000
## N[6]       54.169   6.226  42.529  53.952  67.158 1.001  6600
## N[7]       80.200   7.927  65.633  79.886  96.890 1.001 10000
## N[8]      106.291   9.329  88.859 105.946 125.750 1.001 12000
## N[9]      106.322   9.240  88.903 106.079 125.052 1.001 12000
## N[10]     132.763  10.608 113.047 132.370 154.414 1.001 12000
## N[11]     130.523  10.407 110.982 130.323 151.472 1.001 12000
## N[12]     107.078   9.431  89.372 106.755 126.293 1.001 12000
## N[13]     128.721  10.315 109.385 128.442 150.209 1.001 12000
## N[14]     140.346  10.954 119.610 140.011 162.896 1.001  6900
## N[15]     144.384  10.953 123.789 144.099 166.681 1.001 12000
## N[16]     175.779  12.387 152.613 175.452 201.208 1.001 12000
## N[17]     206.422  13.591 180.667 206.024 234.065 1.001 12000
## N[18]     184.527  13.060 159.709 184.196 211.046 1.001 11000
## N[19]     343.930  18.242 309.177 343.443 380.840 1.001 12000
## N[20]     287.889  16.421 256.153 287.702 320.261 1.001 12000
## N[21]     378.591  18.717 343.078 378.318 416.172 1.001 12000
## N[22]     556.409  23.227 512.832 556.052 602.136 1.002  3000
## N[23]     573.952  23.254 529.815 573.486 620.697 1.001 12000
## N[24]     680.123  25.456 630.713 679.892 730.242 1.001 12000
## N[25]     649.580  24.736 601.550 649.464 699.149 1.001 12000
## N[26]     782.687  27.388 730.052 782.513 837.428 1.001 12000
## N[27]     864.724  29.300 808.515 864.352 923.329 1.001 12000
## alpha       0.224   0.075   0.075   0.223   0.375 1.001  7600
## sigmaProc   0.277   0.056   0.185   0.269   0.403 1.001  8200
## tauProc    14.695   5.905   6.155  13.779  29.115 1.001  8200
## deviance  217.109   7.793 203.701 216.470 234.190 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 = 30.4 and DIC = 247.5
## 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 :

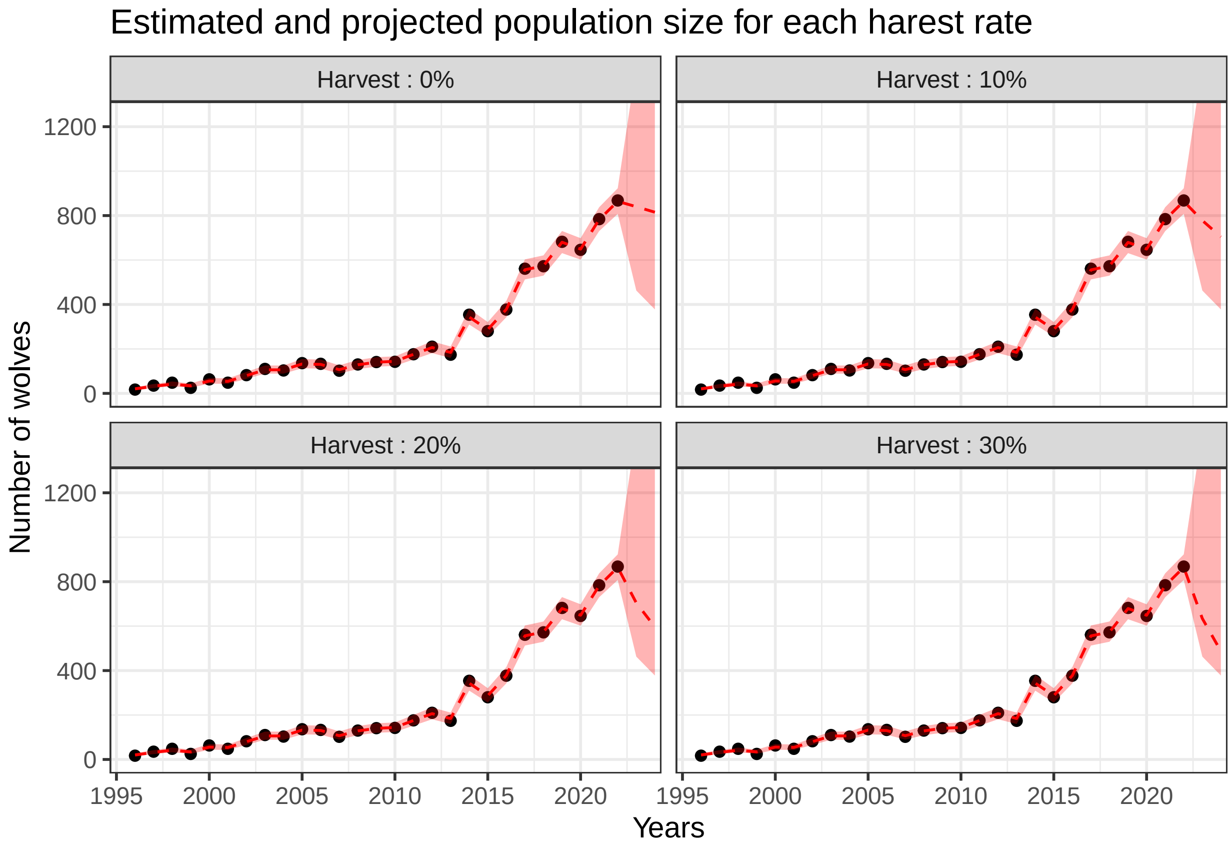
```
for (i in 1:4) {
  # Initialisation des données
  bugs.data = list(
    nyears = nrow(dat),
    y = c(dat$N, rep(NA, 2)),
    dH = dH[i],
    h = dat$H
  )

  # 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

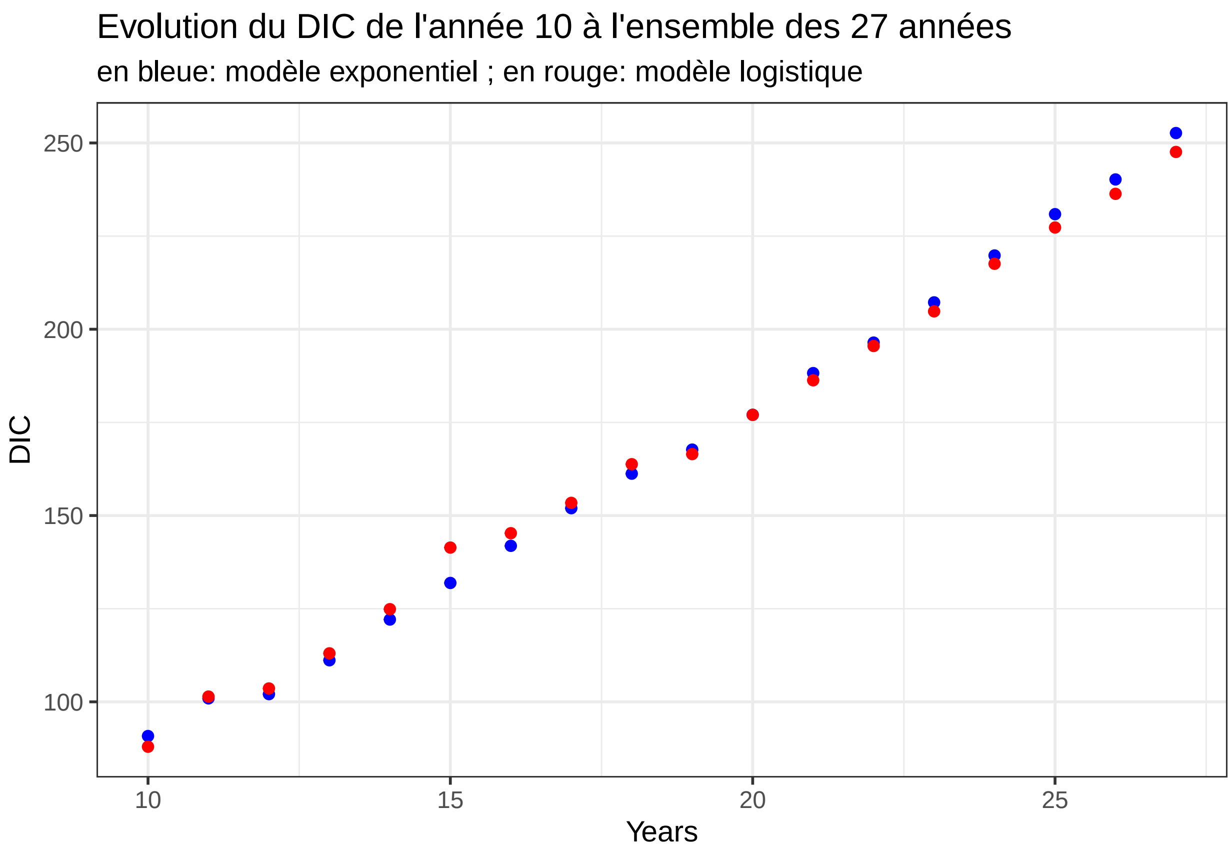
Dans cette section on va comparer l'efficacité de chaque modèle selon le nombre de données, c'est-à-dire en fonction du temps passé.

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,
                   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.

```
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

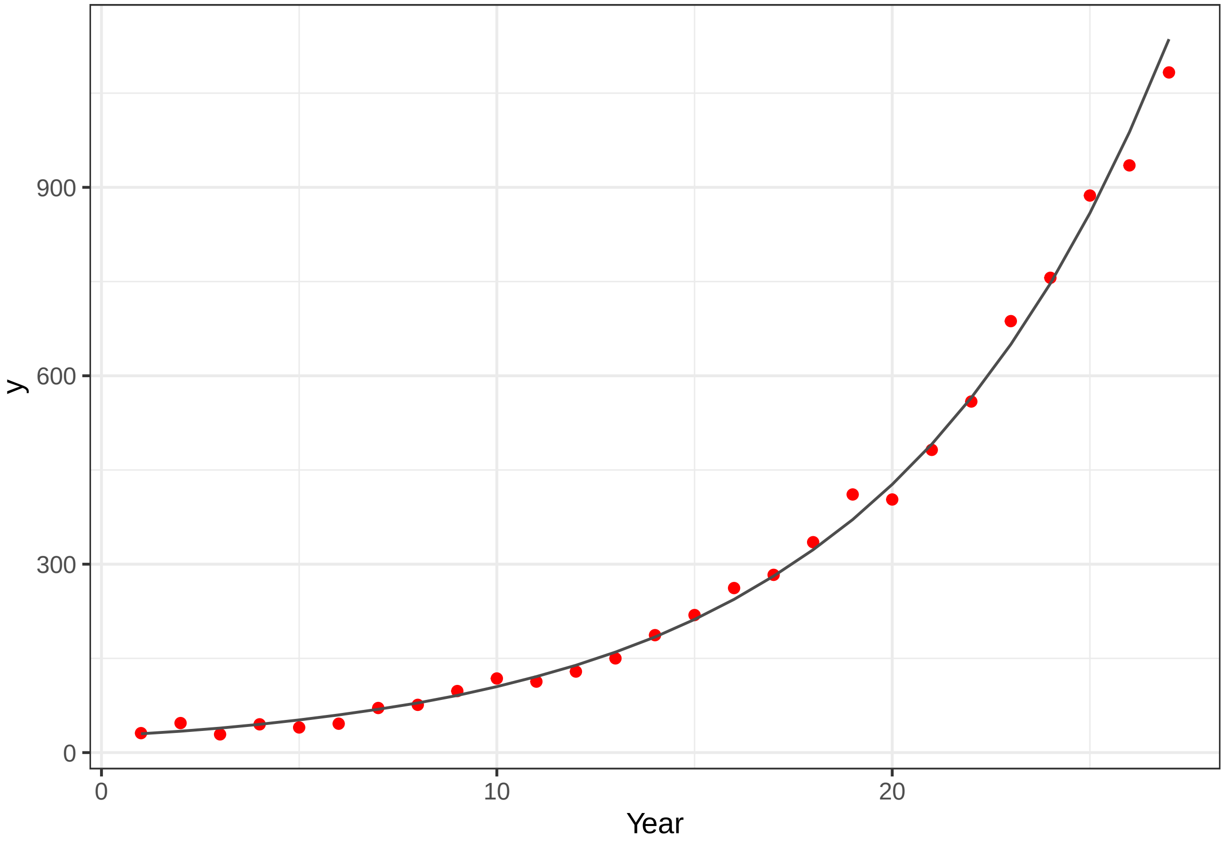
```
nyears = 27
N1 = 30
sigma = 0.15
lambda=1.15
```

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 :

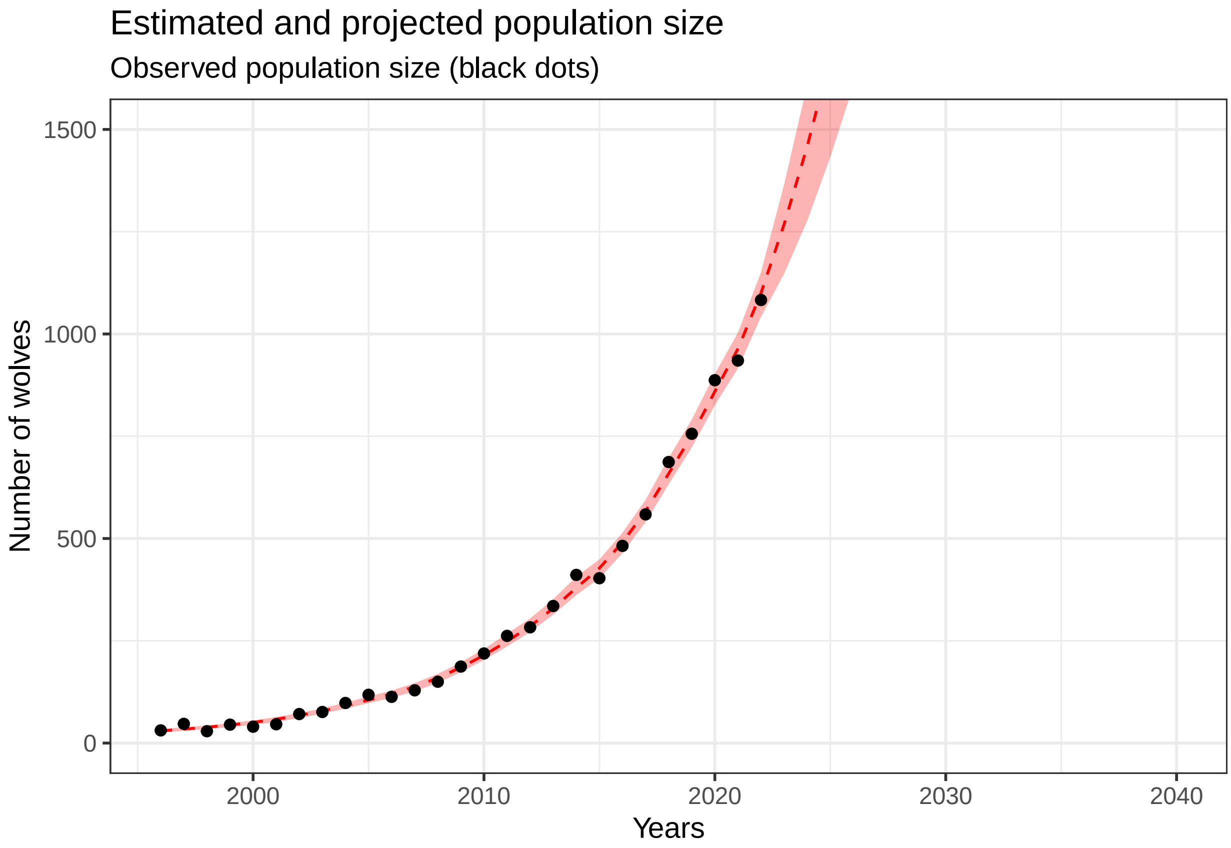
```
bugs.monitor = c("lambda", "sigmaProc", "N", "tauProc")
bugs.chains = 3
bugs.inits = function() {
  list()
}
```

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
```

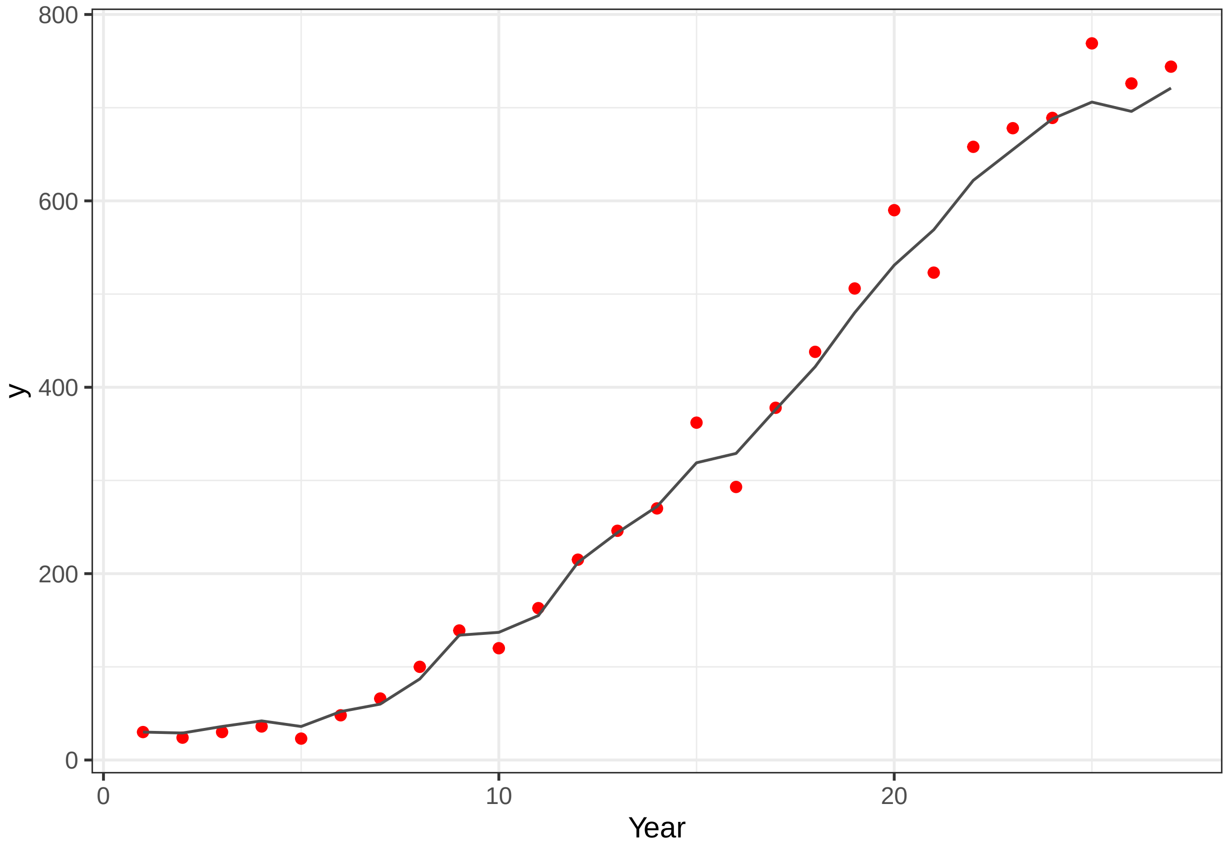
```
sigma = 0.15
K = 800
alpha = 0.2
```

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 :

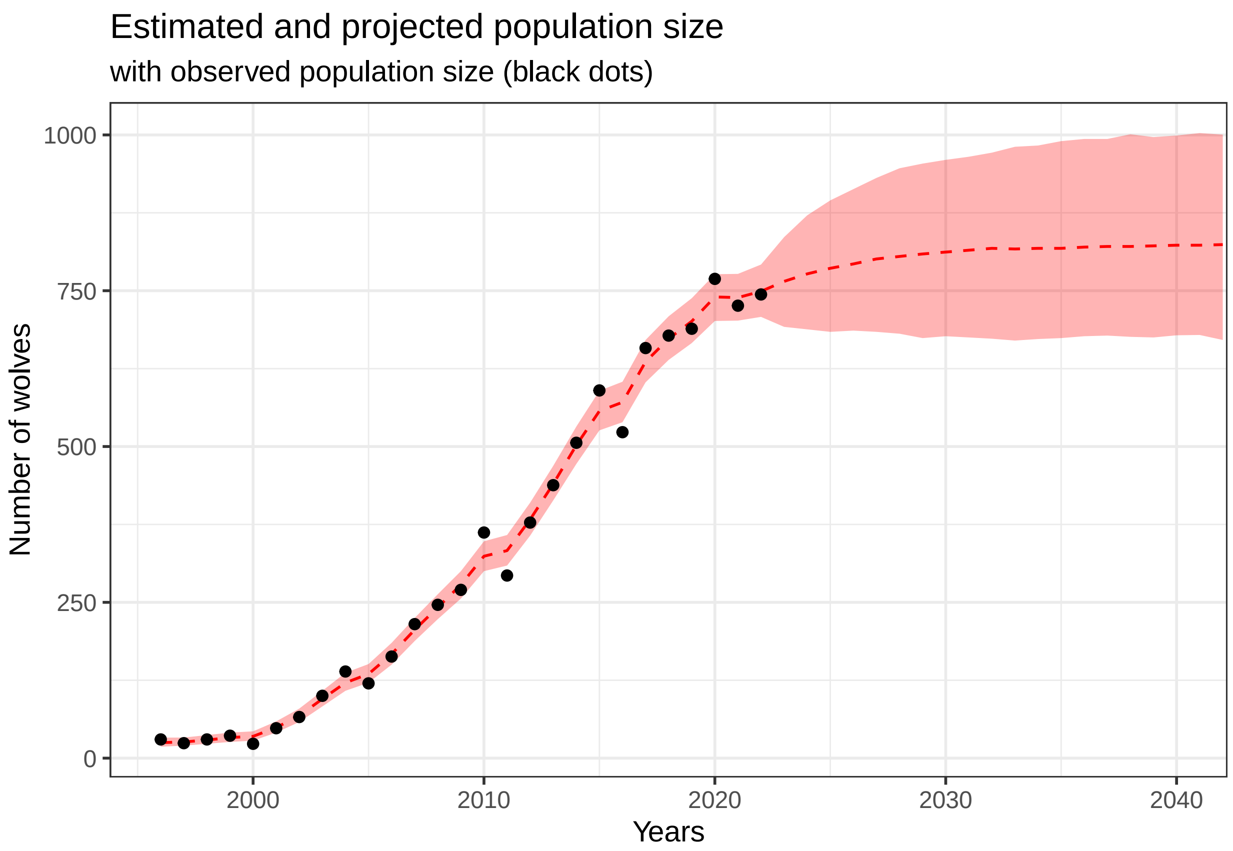
```
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "lambda", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)
```

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

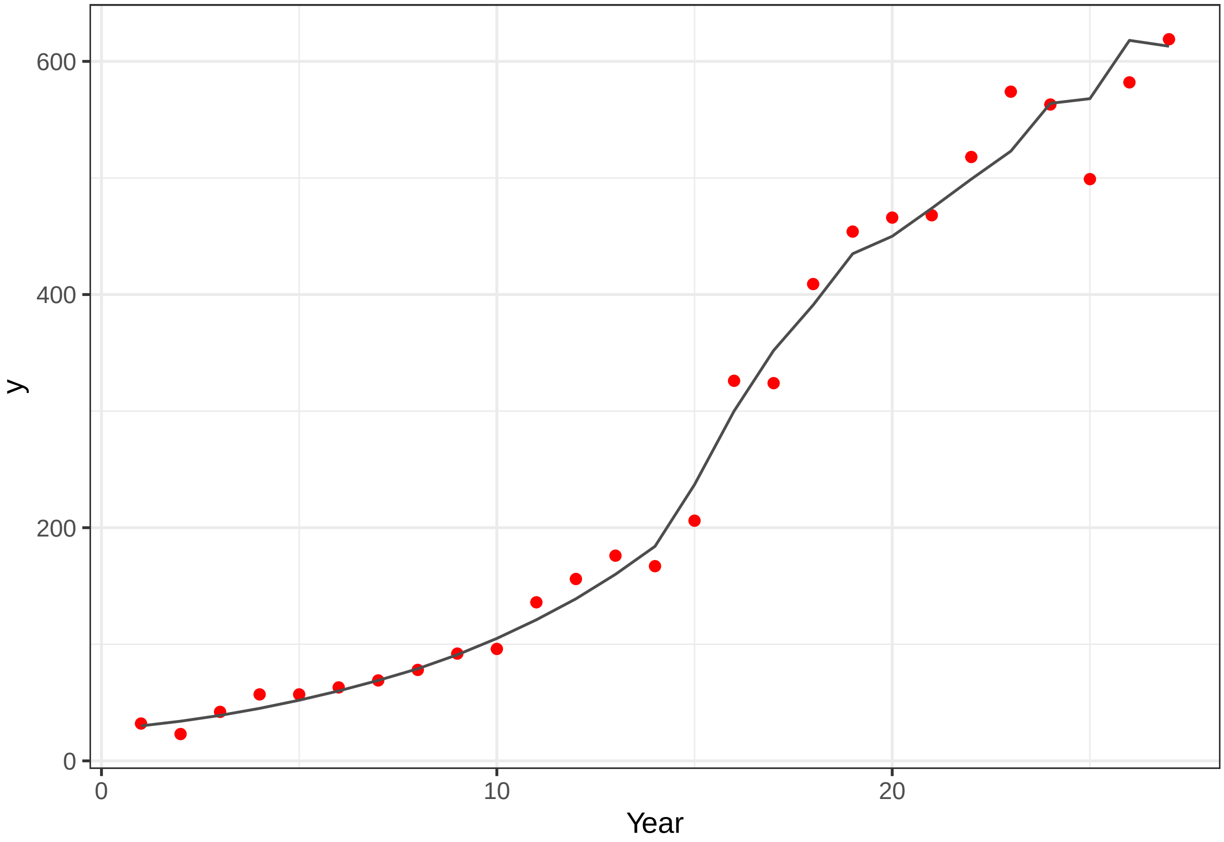
```
nyears = 27
N1 = 30
sigma = 0.15
lambda = 1.15
K = 800
alpha = 0.2
```

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.

```
# 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()
```



```
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 :

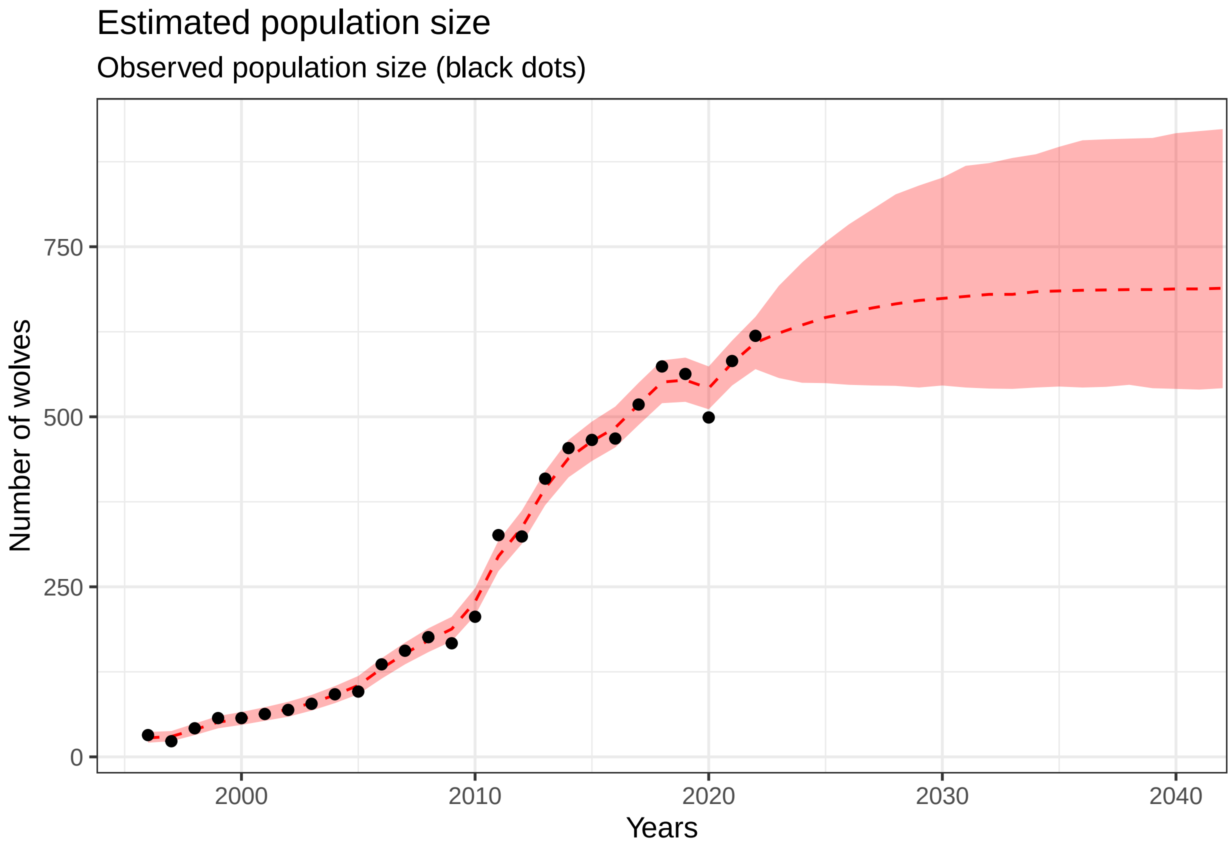
```
bugs.monitor = c("alpha", "sigmaProc", "tauProc", "K", "lambda", "N")
bugs.chains = 3
init1 = list(alpha = .5, sigmaProc = .25)
init2 = list(alpha = .1, sigmaProc = .05)
init3 = list(alpha = 1, sigmaProc = .45)
bugs.inits = list(init1, init2, init3)
```

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.

```r
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 :

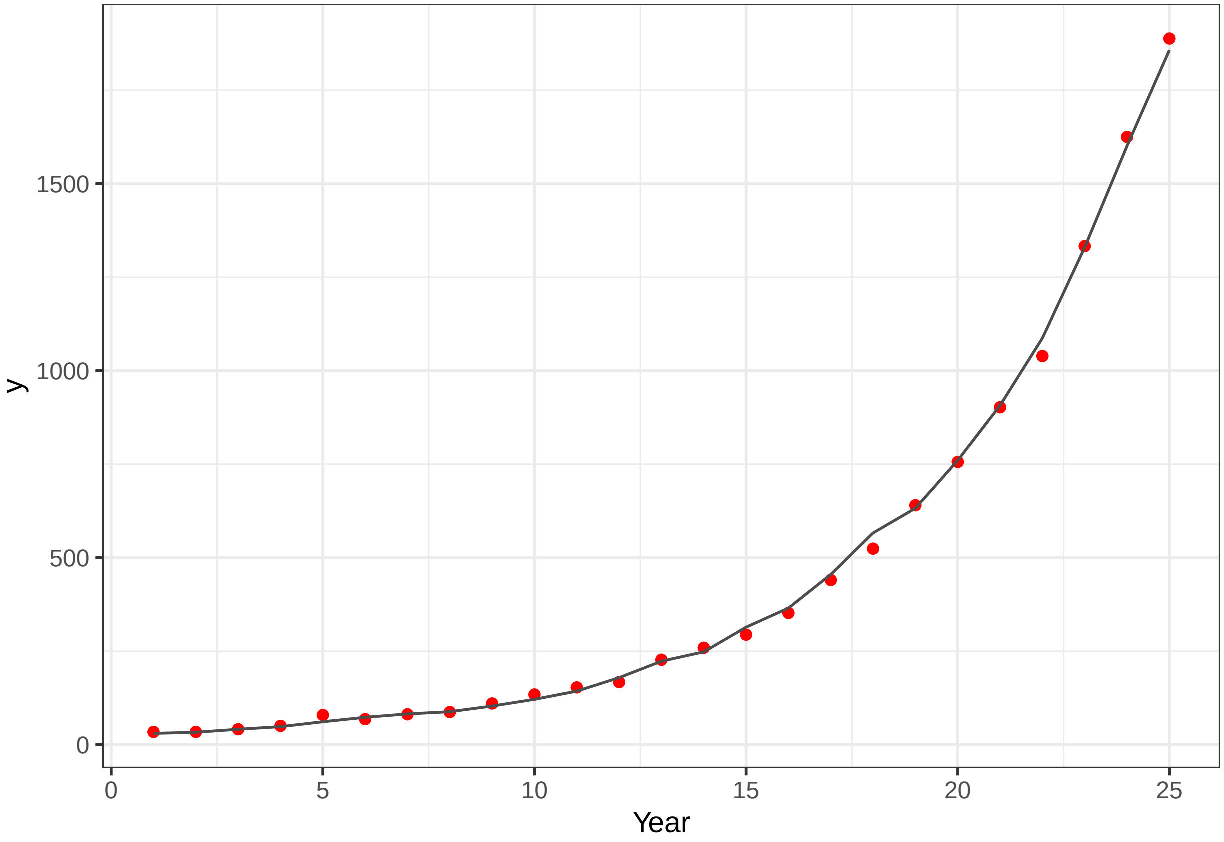
```
dH = c(0,10,20,30)/100
nyears = 25
N1 = 30

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)
}
```

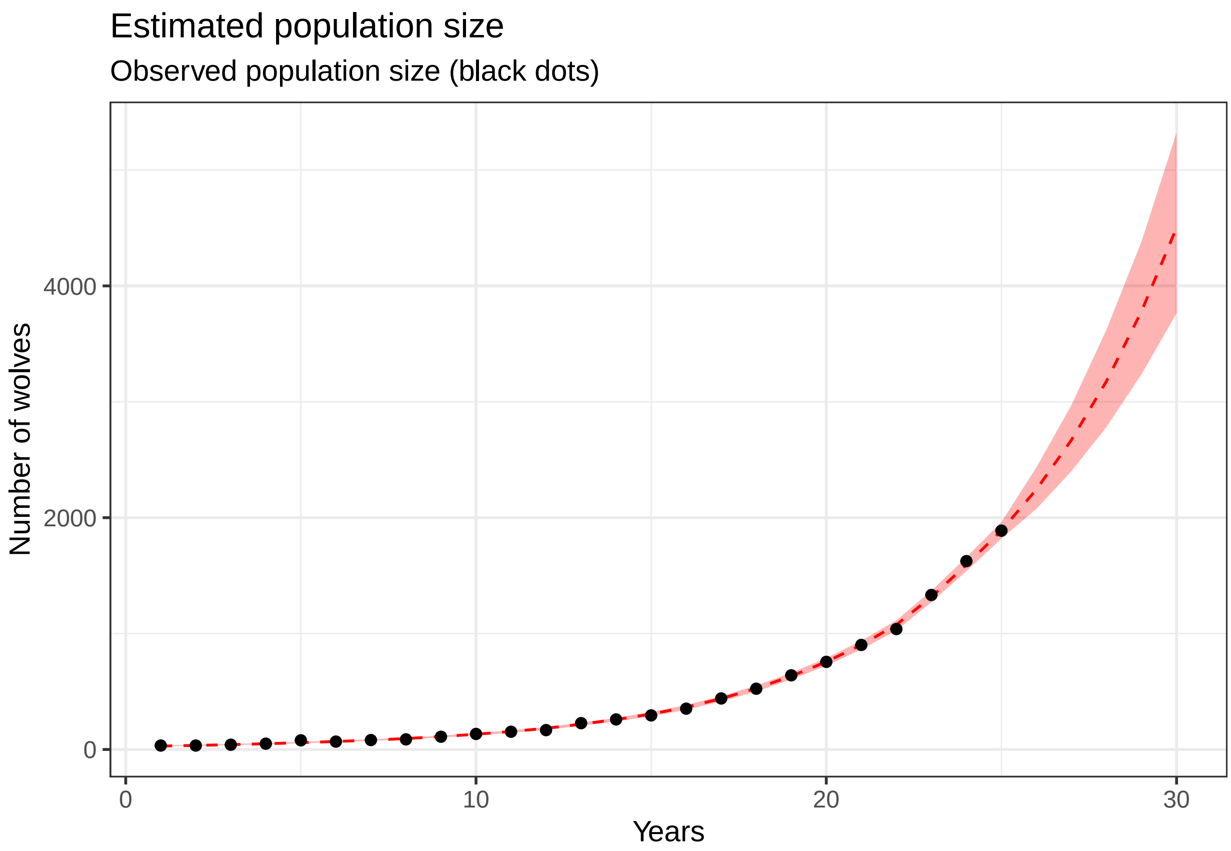
```
ggplot(ssm_sim5, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "grey30") +
  theme_bw()
```



```
tempH
```

```
## [1] 0 0 0 0 0
```

```
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()
```

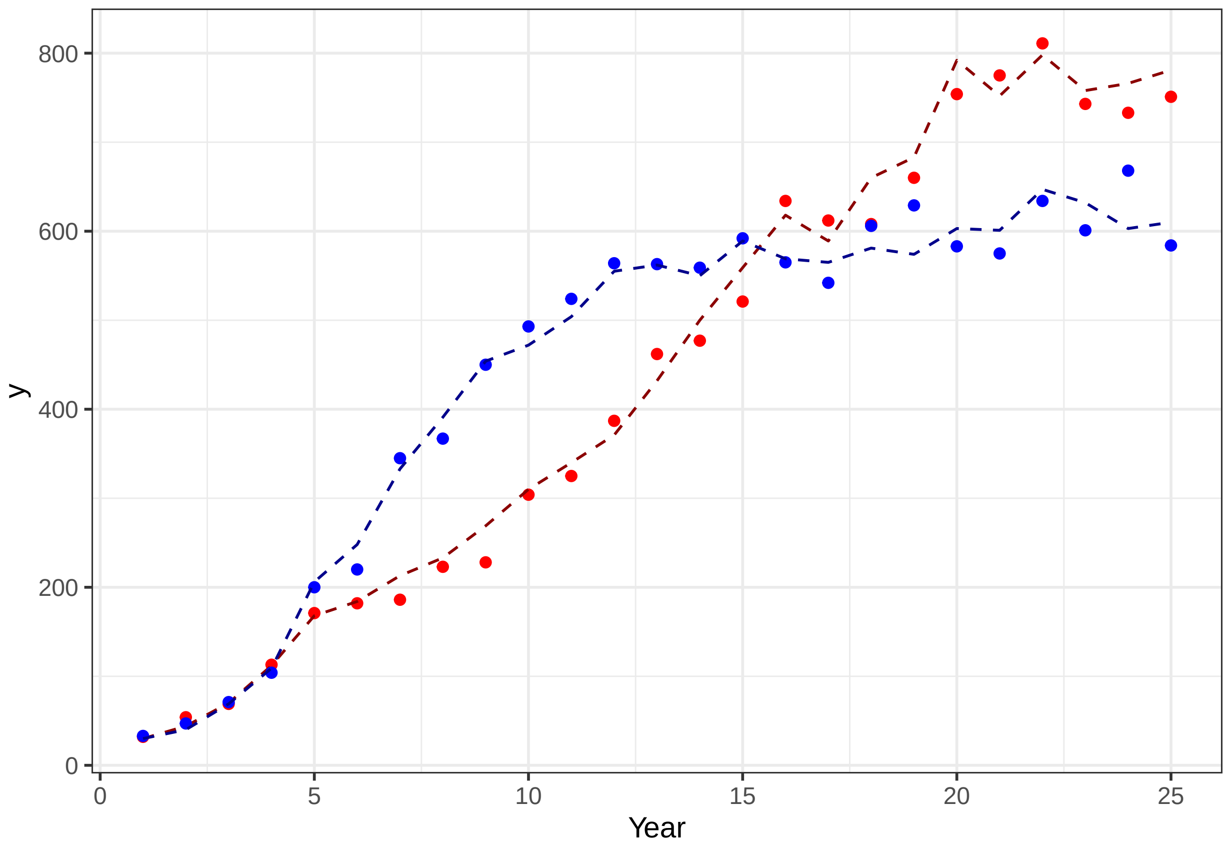
```r
    for (t in 1:(l-5)) {
      lamb[t] = Nest[t+1] / Nest[t]}
    lambda = mean(lamb)
    print(lambda)

    # 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)
}
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 10
##    Total graph size: 72
##
## Initializing model
##
## [1] 1.433336
## [1] 0.3
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3
##    Unobserved stochastic nodes: 11
##    Total graph size: 82
##
## Initializing model
##
## [1] 1.530186
## [1] 0.3
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 4
##    Unobserved stochastic nodes: 12
##    Total graph size: 92
##
## Initializing model
##
## [1] 1.384593
## [1] 0.2
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 13
##    Total graph size: 102
##
## Initializing model
##
## [1] 1.457593
## [1] 0.3
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6
##    Unobserved stochastic nodes: 14
##    Total graph size: 112
##
## Initializing model
##
## [1] 1.456533
## [1] 0.3
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 15
##    Total graph size: 122
##
## Initializing model
##
## [1] 1.378224
## [1] 0.2
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 16
##    Total graph size: 132
##
## Initializing model
##
## [1] 1.304635
## [1] 0.2
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 9
##    Unobserved stochastic nodes: 17
##    Total graph size: 142
##
## Initializing model
```

```
##
## [1] 1.296273
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 18
##    Total graph size: 152
##
## Initializing model
##
## [1] 1.304399
## [1] 0.2
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 11
##    Unobserved stochastic nodes: 19
##    Total graph size: 162
##
## Initializing model
##
## [1] 1.30438
## [1] 0.2
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 12
##    Unobserved stochastic nodes: 20
##    Total graph size: 172
##
## Initializing model
##
## [1] 1.228712
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 13
##    Unobserved stochastic nodes: 21
##    Total graph size: 182
##
## Initializing model
##
## [1] 1.233542
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 22
##    Total graph size: 192
##
## Initializing model
##
## [1] 1.222099
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 23
##    Total graph size: 202
##
## Initializing model
##
## [1] 1.211473
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 16
##    Unobserved stochastic nodes: 24
##    Total graph size: 212
##
## Initializing model
##
## [1] 1.20349
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 17
##    Unobserved stochastic nodes: 25
##    Total graph size: 222
##
## Initializing model
##
## [1] 1.189952
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 26
##    Total graph size: 232
##
## Initializing model
```

```
##
## [1] 1.187682
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 27
##    Total graph size: 242
##
## Initializing model
##
## [1] 1.178942
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 28
##    Total graph size: 252
##
## Initializing model
##
## [1] 1.174421
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 29
##    Total graph size: 262
##
## Initializing model
##
## [1] 1.162627
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 30
##    Total graph size: 272
##
## Initializing model
##
## [1] 1.159714
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 23
##    Unobserved stochastic nodes: 31
##    Total graph size: 282
##
## Initializing model
##
## [1] 1.158892
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 32
##    Total graph size: 292
##
## Initializing model
##
## [1] 1.142295
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 33
##    Total graph size: 302
##
## Initializing model
##
## [1] 1.132031
## [1] 0
```

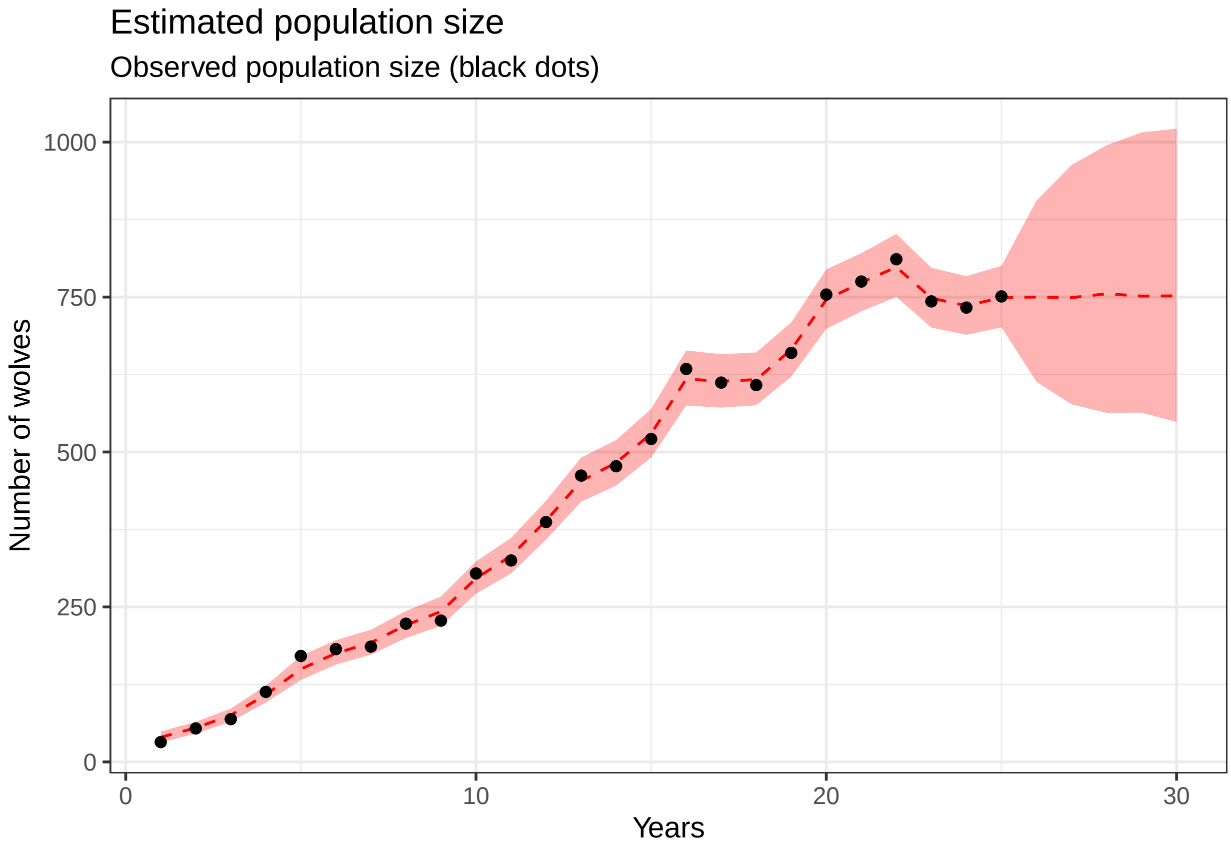
```
ggplot(ssm_sim4, aes(x = Year)) +
  geom_point(aes(y = y), colour = "red") +
  geom_line(aes(y = N), colour = "red4", lty = "dashed") +
  geom_point(aes(y = ybis), colour = "blue") +
  geom_line(aes(y = Nbis), colour = "blue4", lty = "dashed") +
  theme_bw()
```



```
tempH
```

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

```
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")
```



```
tempH
```

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

Comparaison avec une gestion non-adaptative

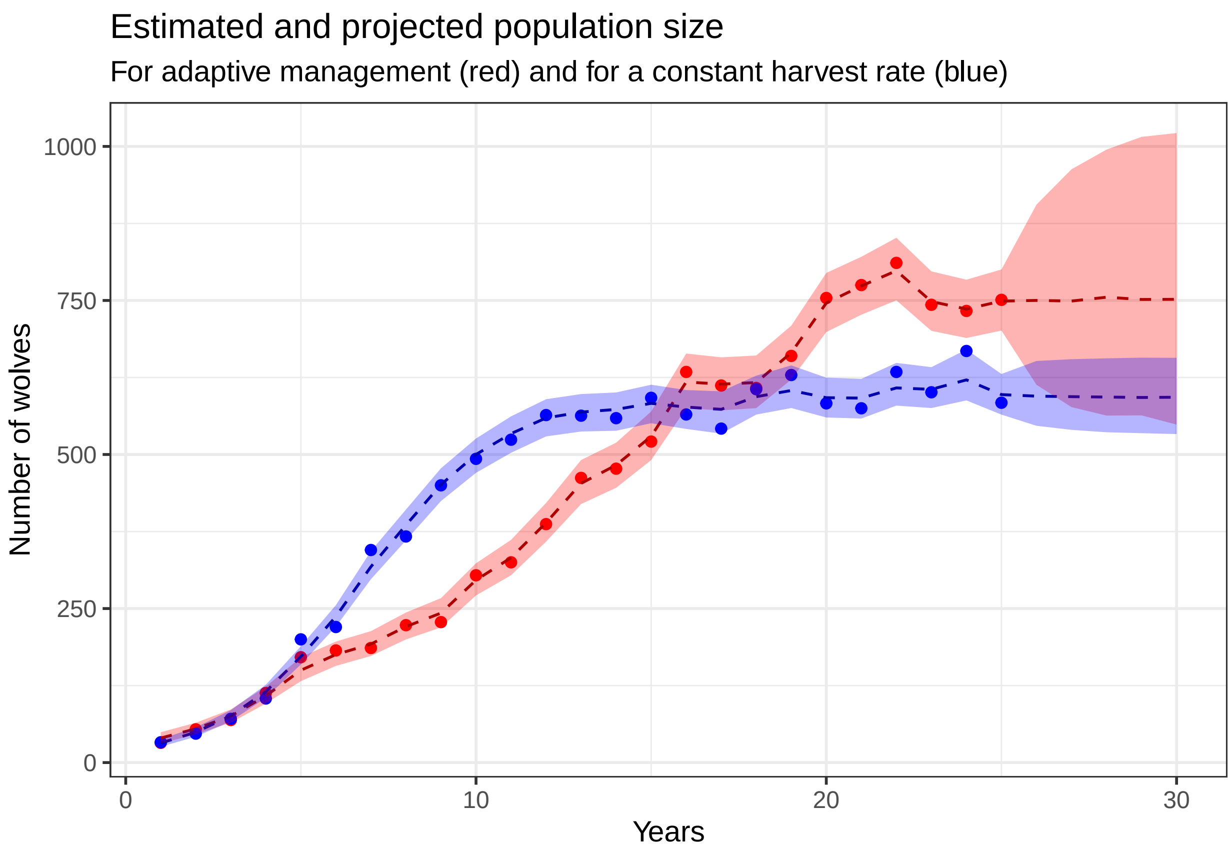
On prend le même jeu de données simulé pécédemment et on applique un taux de prélevement constant chaque année (15% pour coller avec le taux de prélevement observé sur les loups en France).

```
# 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)‘
```

```
  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")
```


Estimated and projected population size
For adaptive management (red) and for a constant harvest rate (blue)
Number of wolves
1000
750
500
250
0
0
10
20
30
Years