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()  %>%
  mutate(se = (CMR/high+low/CMR)/2)
```

```
## Rows: 26 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Years
## dbl (3): low, CMR, high
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
se$se
```

```
##  [1] 0.3048440 0.5863473 0.5973081 0.4111964 0.4902505 0.5645082 0.5769255
##  [8] 0.5941777 0.6035348 0.5975551 0.6135687 0.6138239 0.5992336 0.5994465
## [15] 0.6206024 0.6206336 0.6414100 0.6260278 0.6441957 0.6276282 0.6288178
## [22] 0.7026322 0.7192015 0.7474105 0.7392411 0.8098991
```

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

```
dat <- cbind(round(CMR), c(se$se,0.3), harvest)
colnames(dat) <- c("N", "se", "H")
dat <- as.data.frame(dat)
nyears <- nrow(dat)
dat
```

```
##      N        se   H
## 1   17 0.3048440   0
## 2   35 0.5863473   0
```

```
## 3   48 0.5973081   0
## 4   25 0.4111964   0
## 5   63 0.4902505   0
## 6   48 0.5645082   1
## 7   82 0.5769255   0
## 8  110 0.5941777   0
## 9  103 0.6035348   2
## 10 136 0.5975551   1
## 11 133 0.6135687   2
## 12 102 0.6138239   0
## 13 130 0.5992336   0
## 14 141 0.5994465   1
## 15 142 0.6206024   0
## 16 176 0.6206336   4
## 17 210 0.6414100   4
## 18 174 0.6260278   6
## 19 354 0.6441957  18
## 20 280 0.6276282  36
## 21 377 0.6288178  34
## 22 561 0.7026322  42
## 23 572 0.7192015  51
## 24 682 0.7474105  98
## 25 646 0.7392411 105
## 26 784 0.8098991 103
## 27 868 0.3000000 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.

```
library(R2jags)
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.

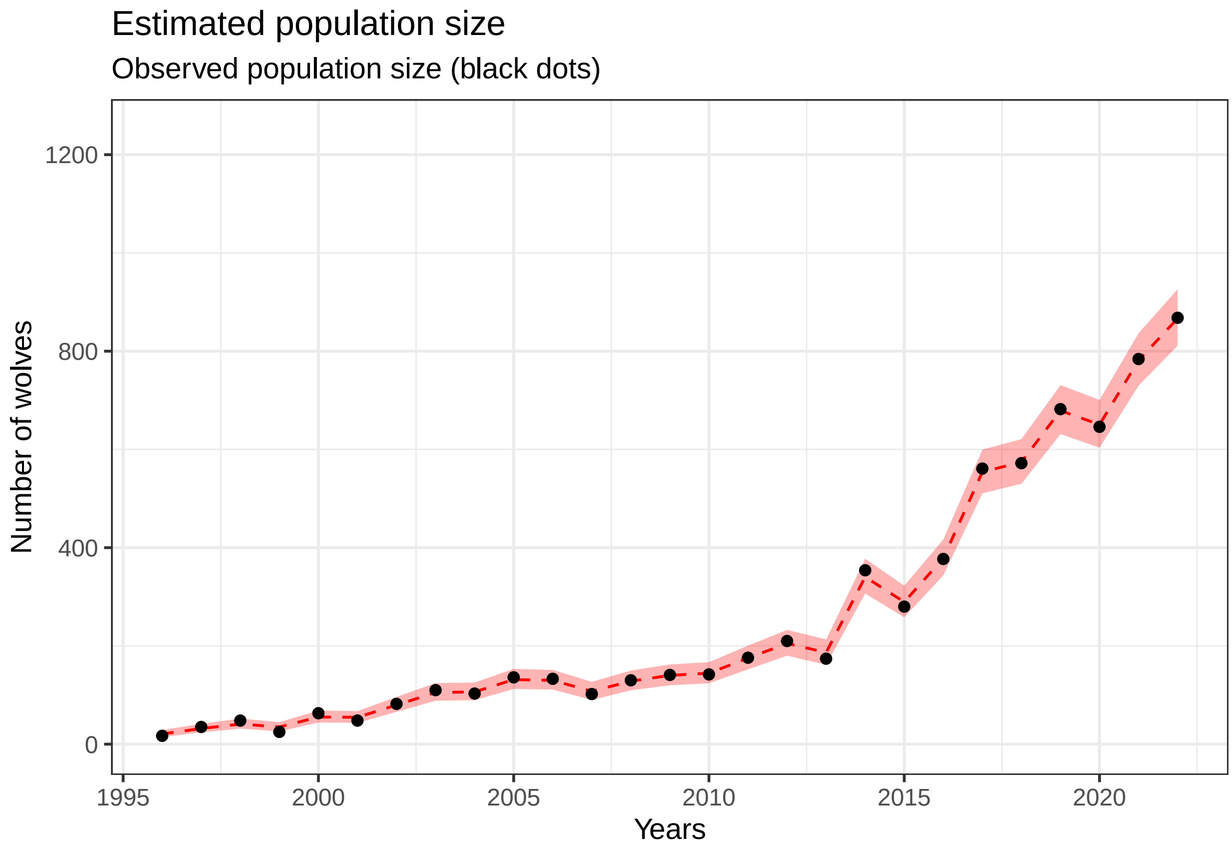
```r
print(wolf_modelexp, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb4909df41.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]       21.013   3.749  14.227  20.793  28.912 1.001 15000
## N[2]       32.021   4.384  24.202  31.769  41.392 1.001 10000
## N[3]       41.148   5.244  31.644  40.846  52.186 1.001 15000
## N[4]       35.148   4.753  26.099  35.040  44.791 1.001  7700
## N[5]       55.405   6.204  44.014  55.119  68.403 1.001  5100
## N[6]       54.893   6.136  43.394  54.717  67.301 1.001 15000
## N[7]       79.959   7.698  65.786  79.614  96.217 1.001 15000
## N[8]      105.470   9.179  88.329 105.107 124.369 1.001 15000
## N[9]      106.714   9.228  89.407 106.405 125.462 1.001  7700
## N[10]     131.822  10.432 112.380 131.413 153.278 1.001 15000
## N[11]     130.133  10.287 111.156 129.812 151.131 1.001 15000
## N[12]     107.888   9.373  90.012 107.665 126.942 1.001  5700
## N[13]     128.612  10.196 109.563 128.306 149.836 1.001 15000
## N[14]     140.287  10.708 119.888 140.006 162.307 1.001 12000
## N[15]     144.650  10.897 123.900 144.490 166.918 1.001 15000
## N[16]     175.526  12.282 152.313 175.167 200.599 1.001 15000
## N[17]     205.434  13.442 180.126 205.000 232.628 1.001  5100
## N[18]     186.809  13.179 161.531 186.543 213.303 1.001 15000
## N[19]     340.799  18.119 306.469 340.433 377.537 1.001 15000
## N[20]     289.630  16.429 258.274 289.346 322.555 1.001 15000
## N[21]     378.893  18.596 343.412 378.575 416.636 1.001 15000
## N[22]     554.430  22.773 510.617 553.924 599.670 1.001 10000
## N[23]     574.669  23.141 530.130 574.481 620.899 1.001  7700
## N[24]     679.309  25.268 631.213 678.892 730.899 1.001 15000
## N[25]     651.011  25.012 603.468 650.556 700.809 1.001 15000
## N[26]     781.924  27.405 729.782 781.249 836.606 1.001 15000
## N[27]     867.361  29.308 811.199 866.644 926.107 1.001 15000
## lambda      1.208   0.062   1.091   1.205   1.337 1.001 15000
## sigmaProc   0.250   0.053   0.162   0.244   0.371 1.001 15000
## tauProc    18.239   7.887   7.275  16.781  37.945 1.001 15000
## deviance  218.858   8.240 204.721 218.179 236.610 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 = 34.0 and DIC = 252.8
## 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.

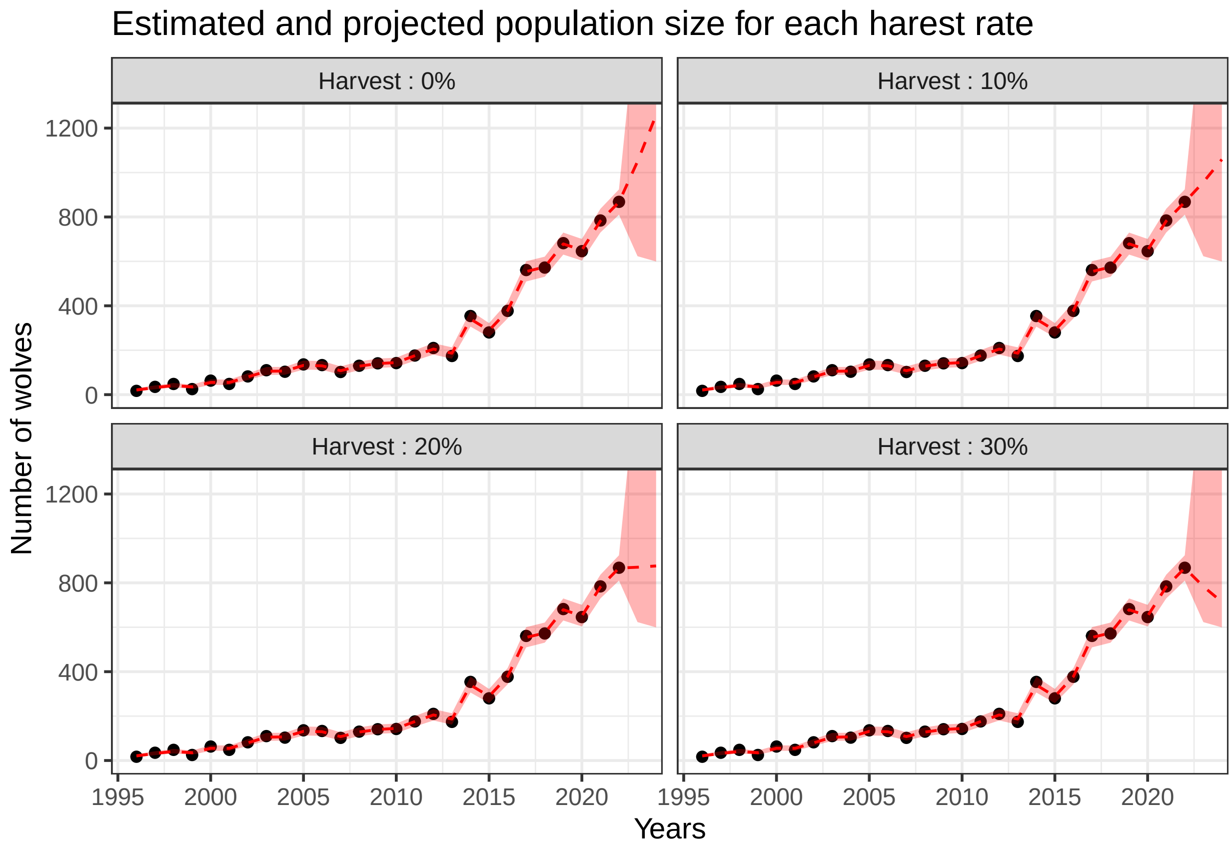
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 31
##    Total graph size: 206
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 31
##    Total graph size: 206
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 31
##    Total graph size: 206
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 31
##    Total graph size: 206
##
## Initializing model
```

On affiche les courbes d'effectifs :

```
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")
```



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.

```
library(R2jags)
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)
```

```
## 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.

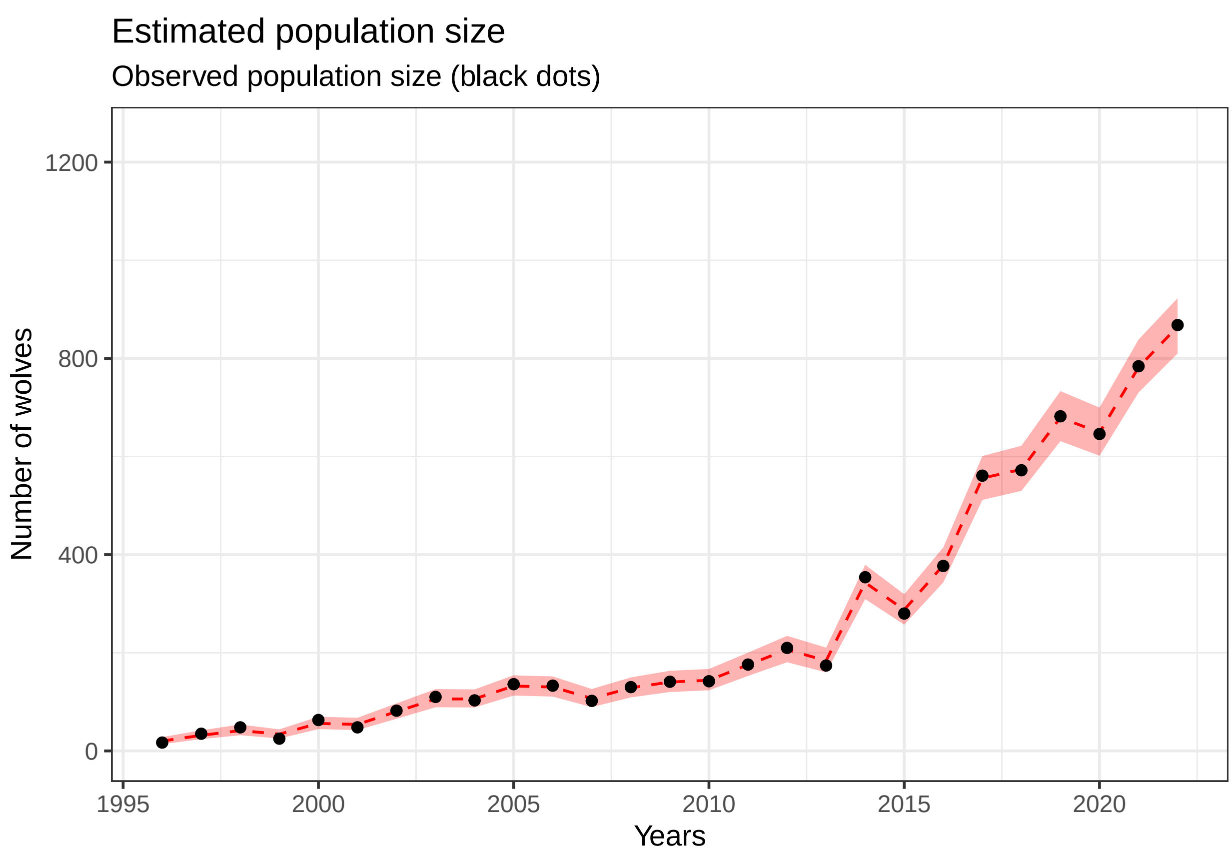
```
print(wolf_modellogist, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb76e95788.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##           mu.vect sd.vect   2.5%     50%   97.5%  Rhat n.eff
## K         782.483 153.278 443.804 806.636 990.890 1.001  4500
## N[1]       20.344   3.708  13.814  20.115  28.230 1.003   760
## N[2]       32.185   4.648  24.049  31.808  42.080 1.002  1900
## N[3]       41.795   5.529  31.955  41.406  53.863 1.001  4500
## N[4]       34.305   4.838  25.273  34.251  43.966 1.001  4500
## N[5]       56.302   6.358  44.452  56.036  69.634 1.001  4500
## N[6]       54.334   6.308  42.479  54.087  67.566 1.003  1000
## N[7]       80.370   8.012  65.487  80.057  96.867 1.002  1700
## N[8]      106.360   9.314  88.928 105.737 125.966 1.001  2600
## N[9]      106.293   9.282  88.609 106.028 125.421 1.001  3300
## N[10]     132.502  10.570 112.835 132.208 154.138 1.001  4500
## N[11]     130.342  10.552 110.739 130.214 151.820 1.001  4500
## N[12]     107.076   9.427  89.341 106.873 126.245 1.001  4500
## N[13]     128.814  10.239 109.024 128.559 149.813 1.002  1100
## N[14]     140.749  10.815 120.036 140.364 163.360 1.001  3600
## N[15]     144.387  11.135 123.598 143.982 167.021 1.002  2300
## N[16]     175.805  12.104 152.761 175.505 200.308 1.002  2100
## N[17]     206.587  13.622 180.733 206.339 234.566 1.001  4500
## N[18]     184.267  12.972 160.120 183.919 210.439 1.002  1100
## N[19]     343.704  17.874 309.458 343.107 379.243 1.001  4500
## N[20]     287.699  16.033 257.315 287.502 319.430 1.001  4500
## N[21]     378.634  18.288 343.437 378.438 414.836 1.001  3700
## N[22]     556.160  23.132 511.510 555.989 601.040 1.001  4500
## N[23]     574.475  23.544 529.991 573.744 621.952 1.001  4500
## N[24]     680.254  25.863 631.441 679.517 733.383 1.001  4000
## N[25]     649.177  24.828 601.627 648.677 699.795 1.001  4500
## N[26]     782.274  27.667 730.422 781.882 838.447 1.001  4500
## N[27]     865.326  29.150 810.091 864.866 922.778 1.001  4500
## alpha       0.222   0.074   0.071   0.223   0.369 1.001  4500
## sigmaProc   0.278   0.056   0.188   0.271   0.408 1.001  4500
## tauProc    14.514   5.737   5.996  13.576  28.150 1.001  4500
## deviance  216.985   7.790 203.515 216.444 234.346 1.001  4500
##
## 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 = 247.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

On affiche la dynamique de la population sur un graphique.

```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) + 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

```
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)
  }
}
```

Initialisation des différents taux de prélèvement :

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

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

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 32
##    Total graph size: 322
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

```
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 32
##    Total graph size: 322
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 32
##    Total graph size: 322
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 32
##    Total graph size: 322
##
## Initializing model
```

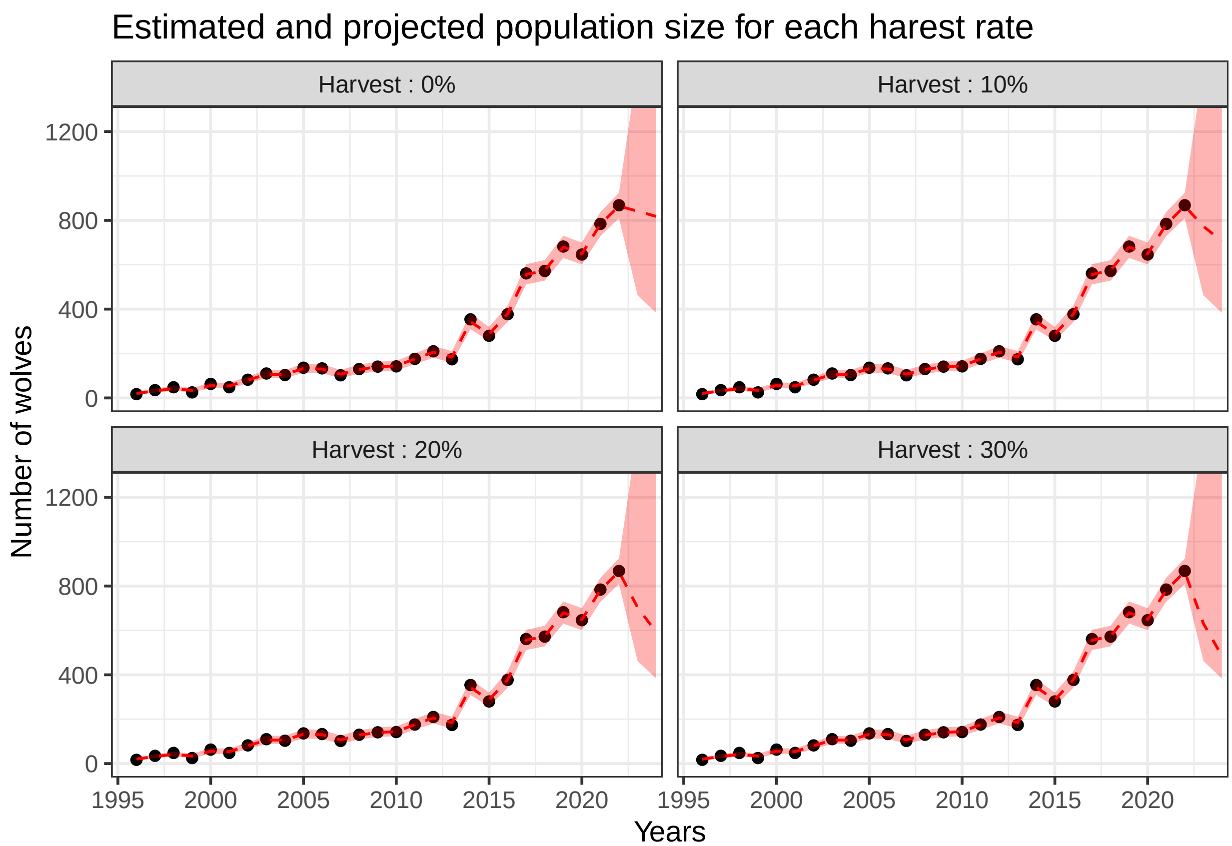
On affiche les estimations et projections pour chaque taux de prélevement :

```
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)) +
```

```r
  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 range le résultat du DIC de la 10ème année jusqu'à la fin.

```r
DICexp=numeric(nrow(dat)-10)
DIClogist=numeric(nrow(dat)-10)
library(R2jags)

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

```r
                 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=100000,
                   n.burnin=50000)

# 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
}
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 12
##    Total graph size: 77
##
```

```
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 13
##    Total graph size: 115
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 11
##    Unobserved stochastic nodes: 13
##    Total graph size: 84
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 11
##    Unobserved stochastic nodes: 14
##    Total graph size: 126
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 12
##    Unobserved stochastic nodes: 14
##    Total graph size: 91
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 12
##    Unobserved stochastic nodes: 15
##    Total graph size: 137
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 13
##    Unobserved stochastic nodes: 15
##    Total graph size: 98
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 13
##    Unobserved stochastic nodes: 16
##    Total graph size: 148
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 16
##    Total graph size: 105
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 17
##    Total graph size: 159
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 17
##    Total graph size: 112
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 18
```

```
##    Total graph size: 170
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 16
##    Unobserved stochastic nodes: 18
##    Total graph size: 119
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 16
##    Unobserved stochastic nodes: 19
##    Total graph size: 181
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 17
##    Unobserved stochastic nodes: 19
##    Total graph size: 126
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 17
##    Unobserved stochastic nodes: 20
##    Total graph size: 192
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 20
##    Total graph size: 133
##
## Initializing model
##
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 21
##    Total graph size: 203
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 21
##    Total graph size: 140
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 22
##    Total graph size: 214
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 22
##    Total graph size: 147
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 23
##    Total graph size: 225
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

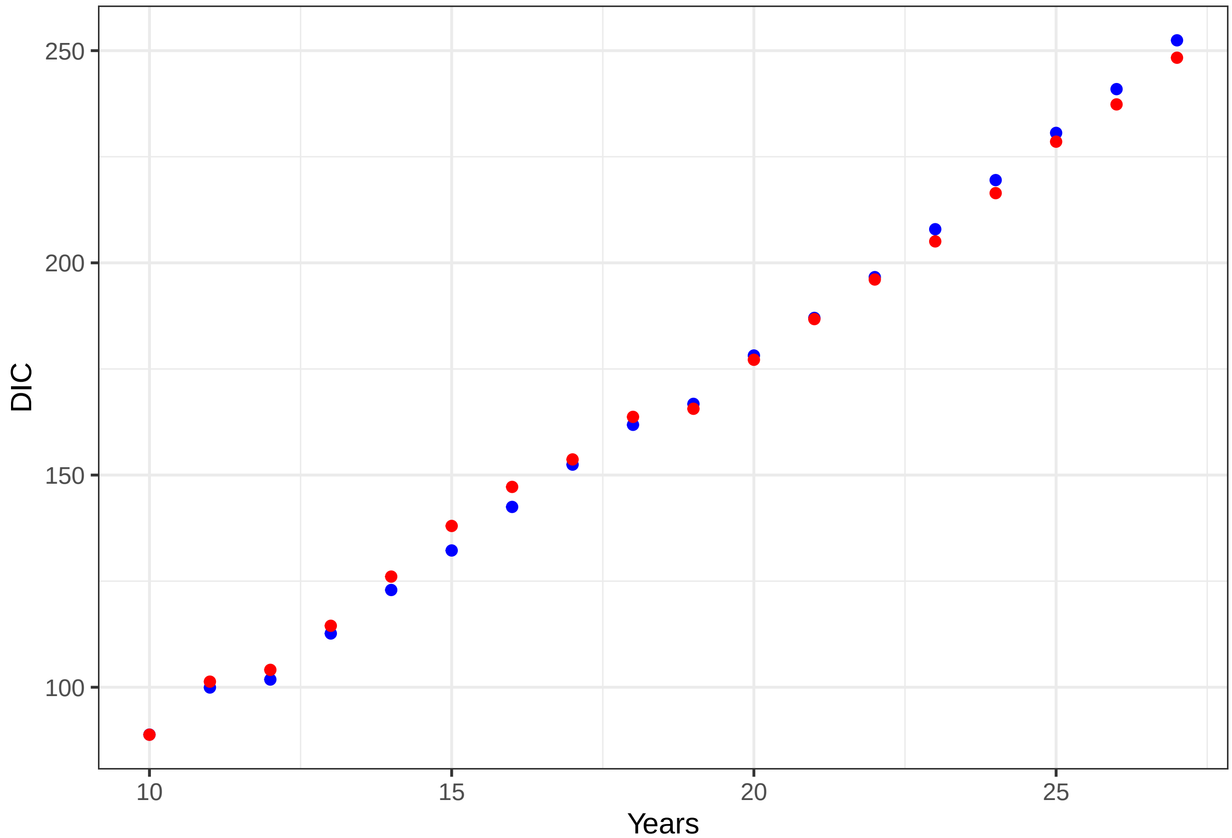
```
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 23
##    Total graph size: 154
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 24
##    Total graph size: 236
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 24
##    Total graph size: 161
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 25
##    Total graph size: 247
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 23
##    Unobserved stochastic nodes: 25
##    Total graph size: 168
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 23
##    Unobserved stochastic nodes: 26
##    Total graph size: 258
##
```

```
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 26
##    Total graph size: 175
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 27
##    Total graph size: 269
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 27
##    Total graph size: 182
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 28
##    Total graph size: 280
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 28
##    Total graph size: 189
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 29
##    Total graph size: 291
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 29
##    Total graph size: 196
##
## Initializing model
##
## 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 l'évolution des DIC des deux modèles au cours du temps.

```r
# DICexp
# DIClogist
ggplot()+
  geom_point(aes(x=seq(10,27),y=DICexp),colour="blue")+
  geom_point(aes(x=seq(10,27),y=DIClogist),colour="red")+
  labs(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

Avec le modèle exponentiel

On initialise les paramètres pour la simulation des données

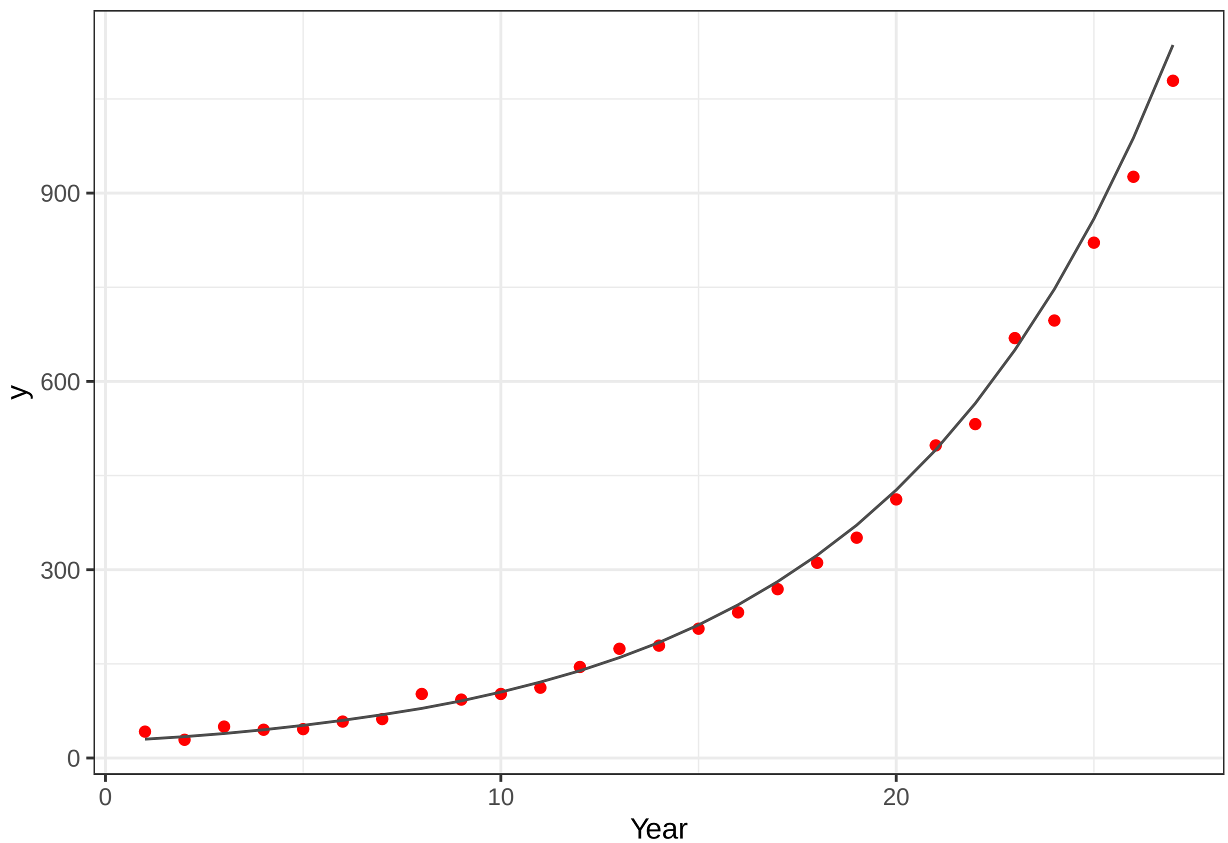
```
nyears = 27
N1 = 30
sigma = 0.15
lambda=1.15
```

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()
}
```

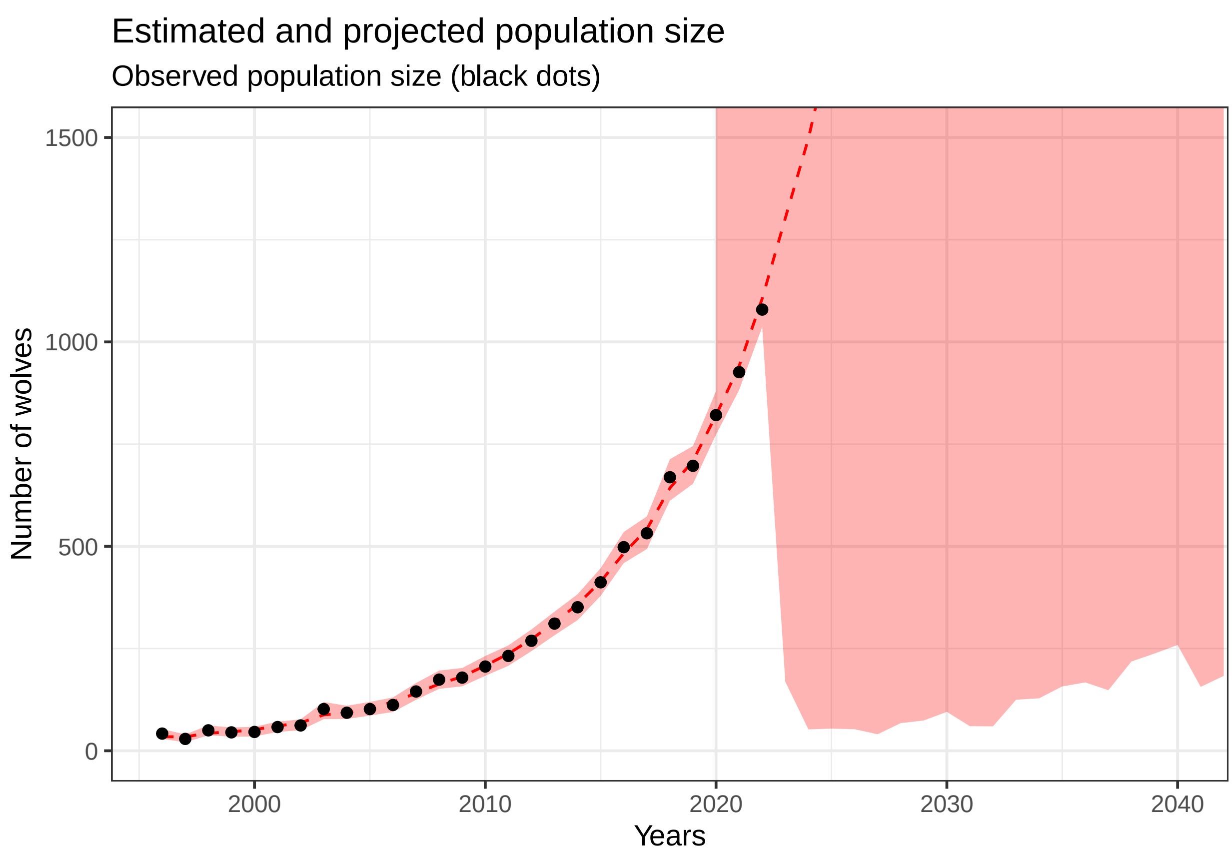
Lancement du modèle.

```
library(R2jags)
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=20000,
                   n.burnin=5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 69
##    Total graph size: 243
##
## Initializing model
```

On affiche la dynamique de la population sur un graphique.

```r
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

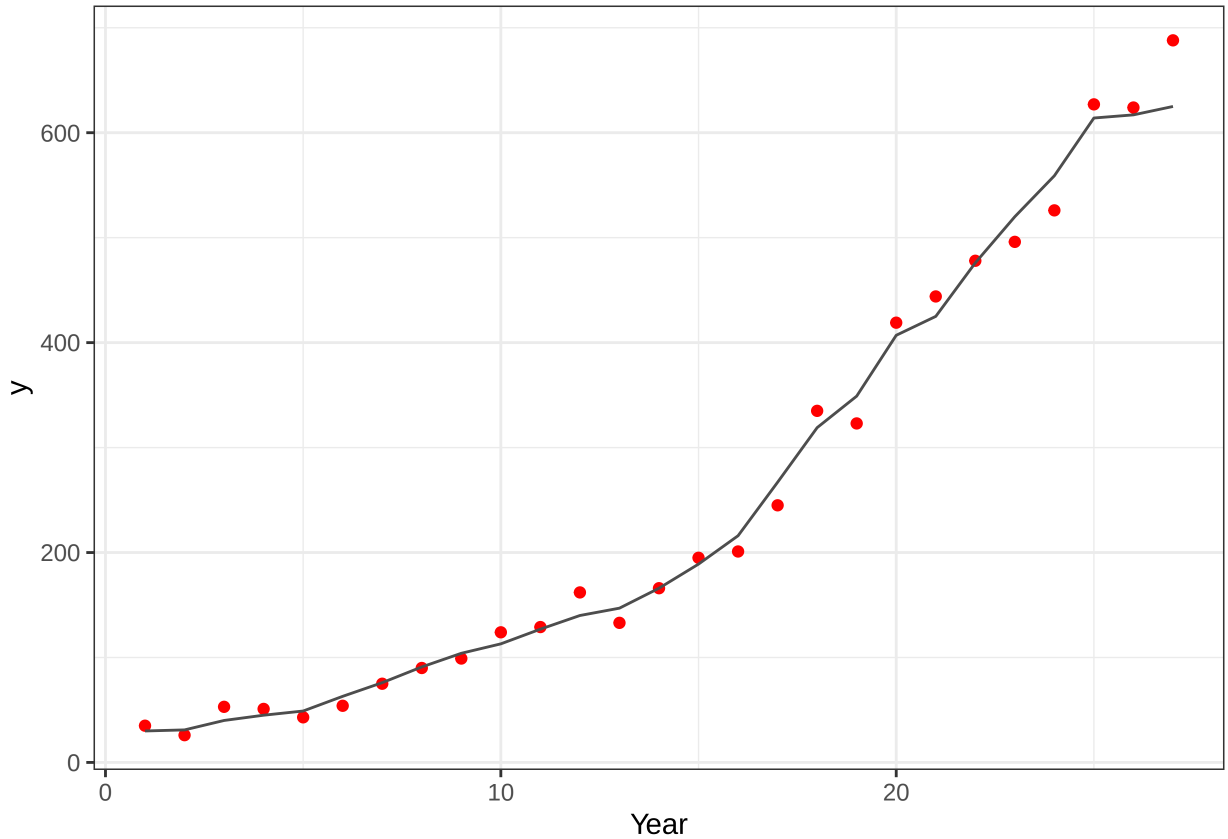
```
nyears = 27
N1 = 30
sigma = 0.15
K = 800
alpha = 0.2
```

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()
```




```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)) {
    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 :

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

Paramètres JAGS :

```r
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.

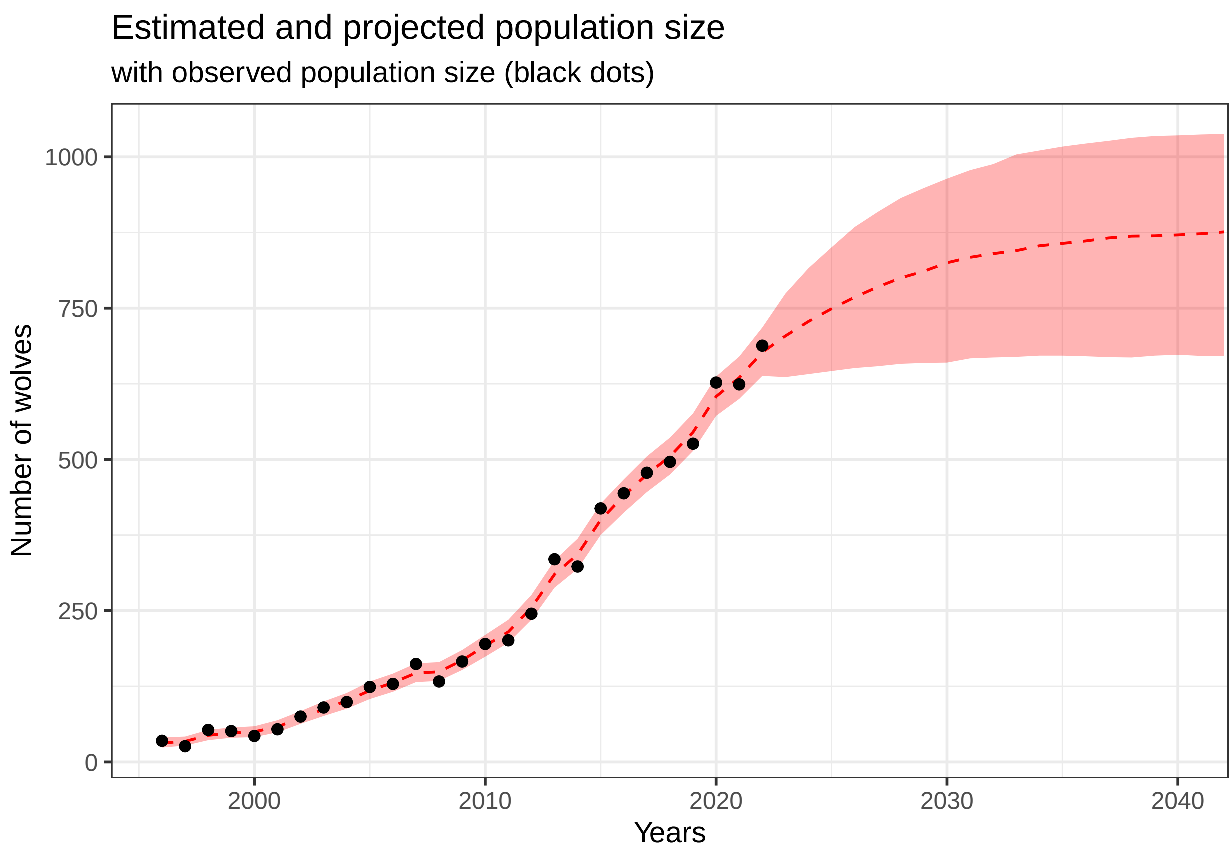
```r
library(R2jags)
sim_modellogist = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modellogist,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=20000,
                   n.burnin=5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 70
##    Total graph size: 337
##
## Initializing model
```

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 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
```

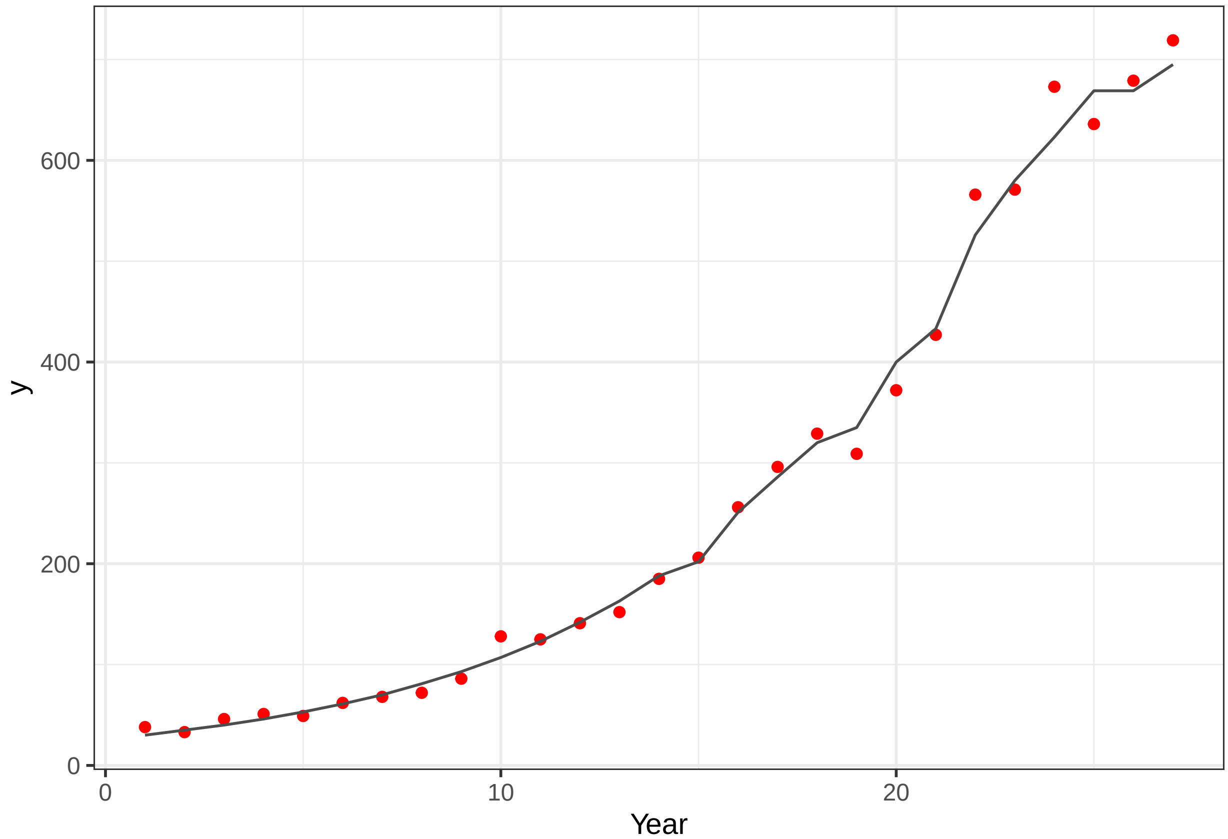
On crée un data frame qui contiendra nos données

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

```r
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()
```




```
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.

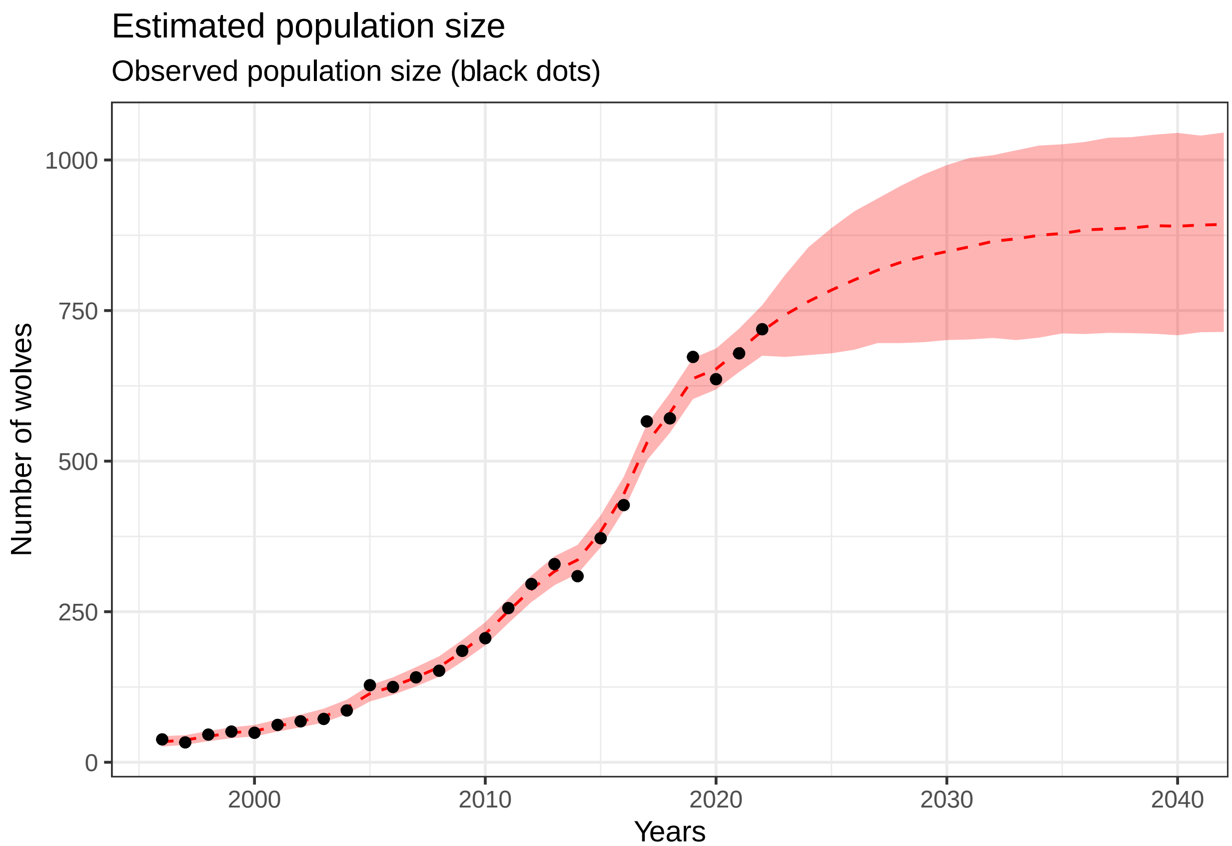
```
library(R2jags)
sim_modellogist = jags(data = bugs.data,
                   inits = bugs.inits,
                   parameters.to.save = bugs.monitor,
                   model.file = modellogist,
                   n.chains = bugs.chains,
                   n.thin=10,
                   n.iter=20000,
                   n.burnin=5000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 27
##    Unobserved stochastic nodes: 70
##    Total graph size: 337
##
## Initializing model
```

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")+
```

```r
  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 logisitique

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

```r
nyears = 25
N1 = 30

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

ssm_sim4$N[1] = N1
```

```r
H = 0
sigma = 0.15
K = 800
alpha = 0.5
ite=0
tempH=c()


for (nyears in seq(5,nyears,5)) { # Boucle sur le nombre d'itérations
  print(H)
  ite=ite+1
  tempH[ite]=H

    if (nyears == 5) {
    for (t in 1:(nyears - 1)) {
      u = ssm_sim4$N[t]*(1-H)
      Er = exp(alpha * (1 - u/ K)) * u
      ssm_sim4$N[t+1] = rpois(1,Er)
    }
  }

  if(nyears>5){
    for (t in (nyears - 5):(nyears - 1)) {
      u = ssm_sim4$N[t]*(1-H)
      Er = exp(alpha * (1 - u/ K)) * u
      ssm_sim4$N[t+1] = rpois(1,Er)
    }
  }

  for (t in 1:nyears){
  ssm_sim4$y[t]=rpois(1,ssm_sim4$N[t])
  }

    # 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,
```

```r
      n.chains = bugs.chains,
      n.thin = 10,
      n.iter = 20000,
      n.burnin = 5000
    )

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

    # Calcul du taux de reproduction estimé
    Nest = wolf_modellogist$BUGSoutput$median$N
    l = length(Nest)
    lamb = c()
    for (t in 1:l-1) {
      lamb[t] = Nest[t+1] / Nest[t]}
    lambda = mean(lamb)
    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)
}
```

```
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 13
##    Total graph size: 102
##
## Initializing model
##
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb15e10ee0.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##            mu.vect   sd.vect    2.5%     50%    97.5%  Rhat n.eff
## K          501.645   261.061 118.596 476.623  968.394 1.001  4500
## N[1]        21.556     4.447  13.658  21.341   30.890 1.001  4500
## N[2]        47.482     6.193  36.708  47.144   60.754 1.001  4500
## N[3]        81.284     8.456  66.149  80.811   98.935 1.001  4500
## N[4]       104.836    10.217  85.472 104.748  124.558 1.002  1600
## N[5]       175.604    12.918 150.616 175.349  201.630 1.001  4500
## N[6]       398.890  6051.021  68.550 236.524  683.214 1.010  4500
## N[7]       360.015   663.006  57.992 292.684  991.157 1.013  3000
## N[8]       410.233   459.230  52.505 332.287 1202.463 1.013  2200
## N[9]       446.609   537.611  48.199 359.238 1329.126 1.014   900
## N[10]      507.765  1228.546  45.628 374.259 1417.084 1.010   920
## alpha        0.637     0.212   0.178   0.631    1.038 1.001  4500
## sigmaProc    0.391     0.378   0.031   0.302    1.287 1.005  1800
```

```
## tauProc   3174.994 163269.810   0.604  10.995 1039.690 1.005  1800
## deviance    35.629      3.196  31.200  35.053   42.996 1.001  4500
##
## 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 = 5.1 and DIC = 40.7
## DIC is an estimate of expected predictive error (lower deviance is better).
## [1] 1.415326
## [1] 0.3
## [1] 0.3
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 18
##    Total graph size: 152
##
## Initializing model
##
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb12054ac6.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##             mu.vect    sd.vect    2.5%     50%     97.5%  Rhat n.eff
## K           282.120    113.078 201.504 247.369   649.216 1.004  3100
## N[1]         25.932      3.808  19.387  25.686    34.155 1.001  4500
## N[2]         45.267      4.541  36.922  45.034    54.772 1.001  4100
## N[3]         71.748      5.921  59.752  71.794    83.447 1.001  4500
## N[4]        108.531      7.884  92.795 108.536   124.123 1.001  2900
## N[5]        159.768     11.479 140.017 158.987   184.537 1.001  2700
## N[6]        177.379     10.407 156.558 177.187   198.147 1.001  2800
## N[7]        186.172     11.809 162.021 186.652   208.011 1.003   900
## N[8]        199.434     11.558 176.815 199.862   222.074 1.002  2200
## N[9]        210.288     11.484 187.384 210.250   232.727 1.002  2200
## N[10]       230.084     15.168 203.403 229.432   260.867 1.002  2000
## N[11]       231.369     41.746 171.517 223.207   335.451 1.001  4500
## N[12]       233.664     56.717 165.030 221.326   378.030 1.001  4500
## N[13]       235.565     65.875 161.492 220.978   402.828 1.002  4500
## N[14]       237.320     74.714 158.639 220.726   427.061 1.004  3900
## N[15]       238.201     75.820 159.000 220.475   442.386 1.003  2200
## alpha         0.938      0.108   0.677   0.959     1.087 1.001  3900
## sigmaProc     0.107      0.076   0.007   0.093     0.294 1.006   540
## tauProc   24083.481 670134.077  11.595 114.589 22813.570 1.006   540
## deviance     75.376      3.792  69.220  75.023    83.578 1.002  1400
##
## 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 = 7.2 and DIC = 82.6
## DIC is an estimate of expected predictive error (lower deviance is better).
## [1] 1.190469
```

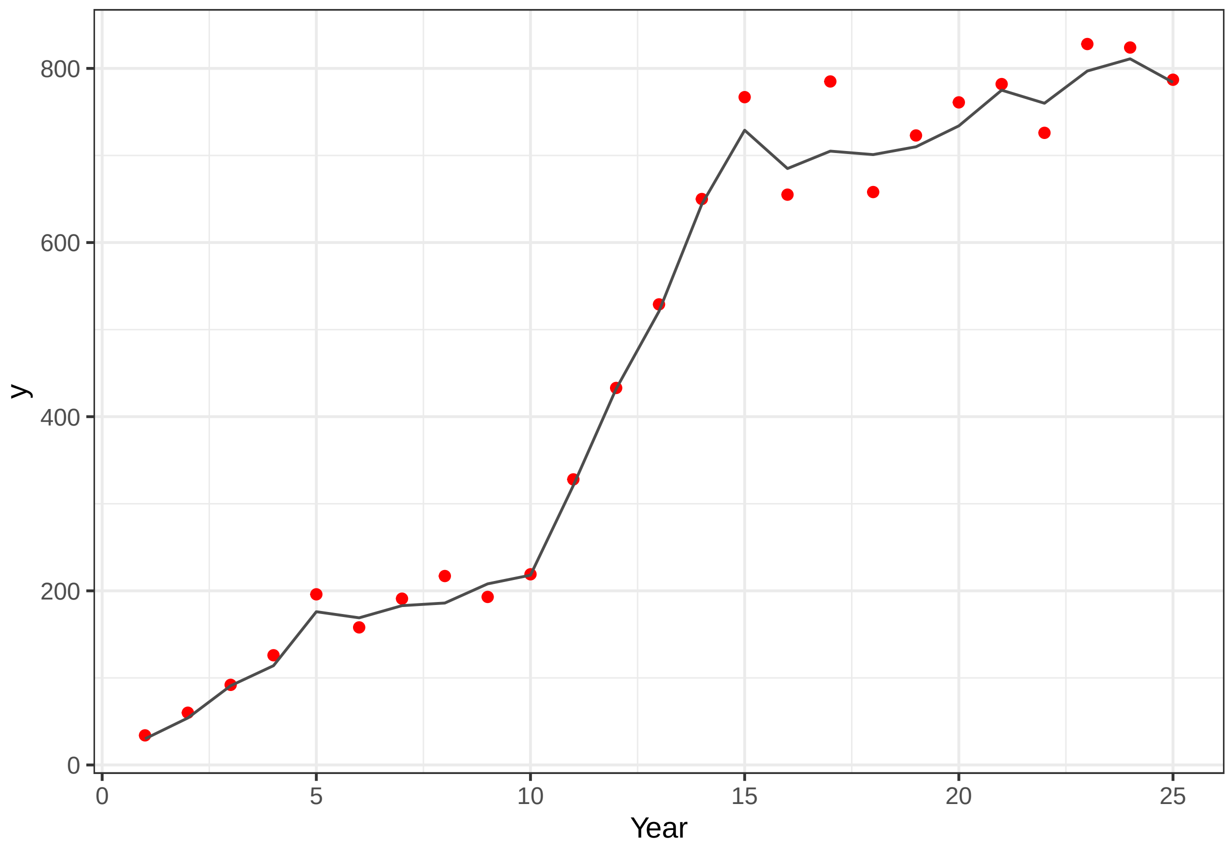
```
## [1] 0
## [1] 0
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 23
##    Total graph size: 202
##
## Initializing model
##
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb75fbf247.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##           mu.vect sd.vect    2.5%     50%    97.5%  Rhat n.eff
## K         699.406 169.570 390.029 698.509  981.813 1.001  4500
## N[1]       24.598   4.487  16.376  24.438   33.990 1.001  4500
## N[2]       46.606   5.624  35.962  46.400   58.259 1.001  4500
## N[3]       90.941   8.618  75.178  90.501  109.464 1.001  2800
## N[4]      125.667  10.120 107.460 125.333  147.303 1.001  4500
## N[5]      193.006  13.345 167.922 192.636  220.017 1.001  4500
## N[6]      167.880  12.113 144.755 167.532  192.170 1.001  3900
## N[7]      185.913  12.718 162.416 185.421  212.138 1.001  2700
## N[8]      182.304  12.922 158.207 182.161  207.574 1.001  4500
## N[9]      215.034  14.099 188.514 214.749  244.023 1.001  4500
## N[10]     217.320  14.002 190.699 217.042  245.502 1.001  4500
## N[11]     309.052  16.782 276.792 309.007  342.743 1.001  4500
## N[12]     427.411  19.997 389.928 427.391  466.188 1.001  4500
## N[13]     533.093  22.654 490.168 532.952  578.607 1.001  4500
## N[14]     620.817  23.874 575.133 620.039  669.361 1.001  3100
## N[15]     724.404  26.492 673.164 723.840  777.471 1.001  4500
## N[16]     725.922 221.405 368.049 696.233 1243.284 1.001  4500
## N[17]     727.359 277.582 319.928 685.438 1380.992 1.001  4500
## N[18]     718.297 296.276 289.567 672.452 1414.833 1.001  4500
## N[19]     717.407 324.853 274.893 664.621 1485.368 1.001  4500
## N[20]     714.316 398.871 275.983 662.721 1447.888 1.001  2600
## alpha       0.370   0.115   0.156   0.363    0.610 1.004  4200
## sigmaProc   0.268   0.071   0.164   0.258    0.439 1.002  1900
## tauProc    16.685   8.332   5.197  15.077   37.308 1.002  1900
## deviance  121.444   5.428 112.787 120.799  133.839 1.001  4500
##
## 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 = 14.7 and DIC = 136.2
## DIC is an estimate of expected predictive error (lower deviance is better).
## [1] 1.221949
## [1] 0.1
## [1] 0.1
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

```
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 28
##    Total graph size: 252
##
## Initializing model
##
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb63dca408.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##           mu.vect sd.vect    2.5%     50%    97.5%  Rhat n.eff
## K         835.241 107.662 605.018 848.751  991.197 1.001  4500
## N[1]       36.752   5.241  26.743  36.622   47.468 1.001  3600
## N[2]       64.166   6.549  52.094  63.871   77.626 1.001  4500
## N[3]       90.054   7.993  75.441  89.714  107.050 1.002  1700
## N[4]       97.872   8.680  80.898  97.940  115.708 1.001  3200
## N[5]      170.884  12.460 147.749 170.823  196.246 1.002  1600
## N[6]      171.571  11.446 150.141 171.380  194.564 1.002  2300
## N[7]      178.260  11.619 156.480 177.973  201.746 1.002  2100
## N[8]      185.460  12.258 162.553 185.196  209.893 1.001  4500
## N[9]      187.384  12.572 163.223 187.219  212.376 1.002  2200
## N[10]     236.471  13.813 210.681 236.102  264.768 1.001  4500
## N[11]     306.920  16.264 276.879 306.509  339.425 1.001  3400
## N[12]     437.480  19.641 399.870 437.030  476.519 1.002  1500
## N[13]     533.014  22.406 489.482 532.989  577.193 1.001  4500
## N[14]     620.923  24.156 574.813 620.761  669.568 1.001  3100
## N[15]     694.706  25.259 646.408 694.450  745.199 1.001  4500
## N[16]     748.993  26.970 698.028 748.417  803.962 1.002  1200
## N[17]     666.453  24.884 617.248 666.481  714.586 1.001  4100
## N[18]     690.544  24.978 643.700 690.072  740.159 1.001  4500
## N[19]     707.972  25.486 659.486 707.182  760.273 1.001  4500
## N[20]     751.719  26.817 699.929 751.550  806.001 1.001  4500
## N[21]     742.831 155.566 482.805 730.242 1088.438 1.001  4500
## N[22]     730.003 186.950 414.314 716.381 1152.580 1.001  4500
## N[23]     717.108 196.381 382.470 701.054 1151.391 1.001  4500
## N[24]     711.703 202.039 367.016 693.311 1160.823 1.001  4500
## N[25]     707.267 206.255 359.483 688.361 1179.744 1.002  1800
## alpha       0.422   0.072   0.288   0.420    0.573 1.002  1400
## sigmaProc   0.191   0.048   0.117   0.185    0.302 1.001  3100
## tauProc    32.793  16.641  10.979  29.270   72.751 1.001  3100
## deviance  172.158   6.813 160.818 171.404  186.935 1.003   840
##
## 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 = 23.2 and DIC = 195.3
## DIC is an estimate of expected predictive error (lower deviance is better).
## [1] 1.148754
## [1] 0
## [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
##
## Inference for Bugs model at "/tmp/Rtmp3d5LnS/model56fb75ea0b08.txt", fit using jags,
##  3 chains, each with 20000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 4500 iterations saved
##           mu.vect sd.vect    2.5%     50%    97.5%  Rhat n.eff
## K         795.831 101.732 604.338 792.835  981.928 1.002  1800
## N[1]       39.949   5.170  30.517  39.740   50.479 1.001  4500
## N[2]       60.153   5.800  49.272  60.010   71.984 1.001  4500
## N[3]       89.720   7.285  76.740  89.441  104.801 1.001  4500
## N[4]      126.342   9.037 109.942 125.825  145.387 1.001  4500
## N[5]      179.096  12.319 156.368 178.779  204.015 1.001  4500
## N[6]      166.667  10.888 146.099 166.553  188.664 1.001  4500
## N[7]      190.426  11.835 168.646 190.415  214.679 1.001  4500
## N[8]      211.369  12.682 187.600 211.054  236.889 1.001  4500
## N[9]      201.449  12.584 177.222 201.384  227.228 1.002  1400
## N[10]     229.063  13.513 203.103 229.005  256.164 1.001  4500
## N[11]     326.060  16.288 295.191 325.916  358.978 1.001  2700
## N[12]     431.236  19.267 394.617 430.718  470.585 1.001  4500
## N[13]     529.545  21.709 487.705 528.843  573.115 1.001  3800
## N[14]     648.512  24.331 602.784 647.853  697.163 1.001  2900
## N[15]     757.198  26.336 707.559 756.394  808.761 1.002  1500
## N[16]     664.705  24.851 617.936 664.478  713.548 1.001  4500
## N[17]     774.651  26.682 721.960 774.790  827.692 1.001  4500
## N[18]     666.692  25.227 617.795 666.369  716.089 1.001  4500
## N[19]     721.910  25.192 672.487 721.838  771.909 1.001  2600
## N[20]     760.212  26.054 709.774 760.054  812.063 1.001  4500
## N[21]     779.440  26.727 727.629 779.309  832.261 1.001  4500
## N[22]     731.989  25.749 682.193 732.074  784.266 1.001  4500
## N[23]     823.554  27.553 768.881 823.408  877.664 1.002  1600
## N[24]     822.036  27.702 768.423 821.931  876.583 1.001  3600
## N[25]     788.279  27.126 735.052 788.024  841.860 1.001  3600
## N[26]     795.324 132.263 564.284 786.539 1087.868 1.001  4500
## N[27]     800.156 166.218 511.843 787.262 1172.855 1.001  3600
## N[28]     797.613 183.065 495.149 782.714 1214.136 1.001  4500
## N[29]     795.367 193.025 476.986 774.438 1239.136 1.002  2400
## N[30]     797.914 201.706 472.240 775.561 1266.190 1.001  4500
## alpha       0.293   0.062   0.174   0.292    0.420 1.001  4500
## sigmaProc   0.155   0.031   0.105   0.151    0.224 1.001  2700
## tauProc    46.802  18.163  20.005  43.910   91.070 1.001  2700
## deviance  217.227   7.184 205.351 216.455  233.390 1.001  2800
##
## 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 = 25.8 and DIC = 243.0
## DIC is an estimate of expected predictive error (lower deviance is better).
## [1] 1.122041
```

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

```
ggplot(ssm_sim4, aes(x=Year))+
  geom_point(aes(y=y),colour="red")+
  geom_line(aes(y=N),colour="grey30")+
  theme_bw()
```

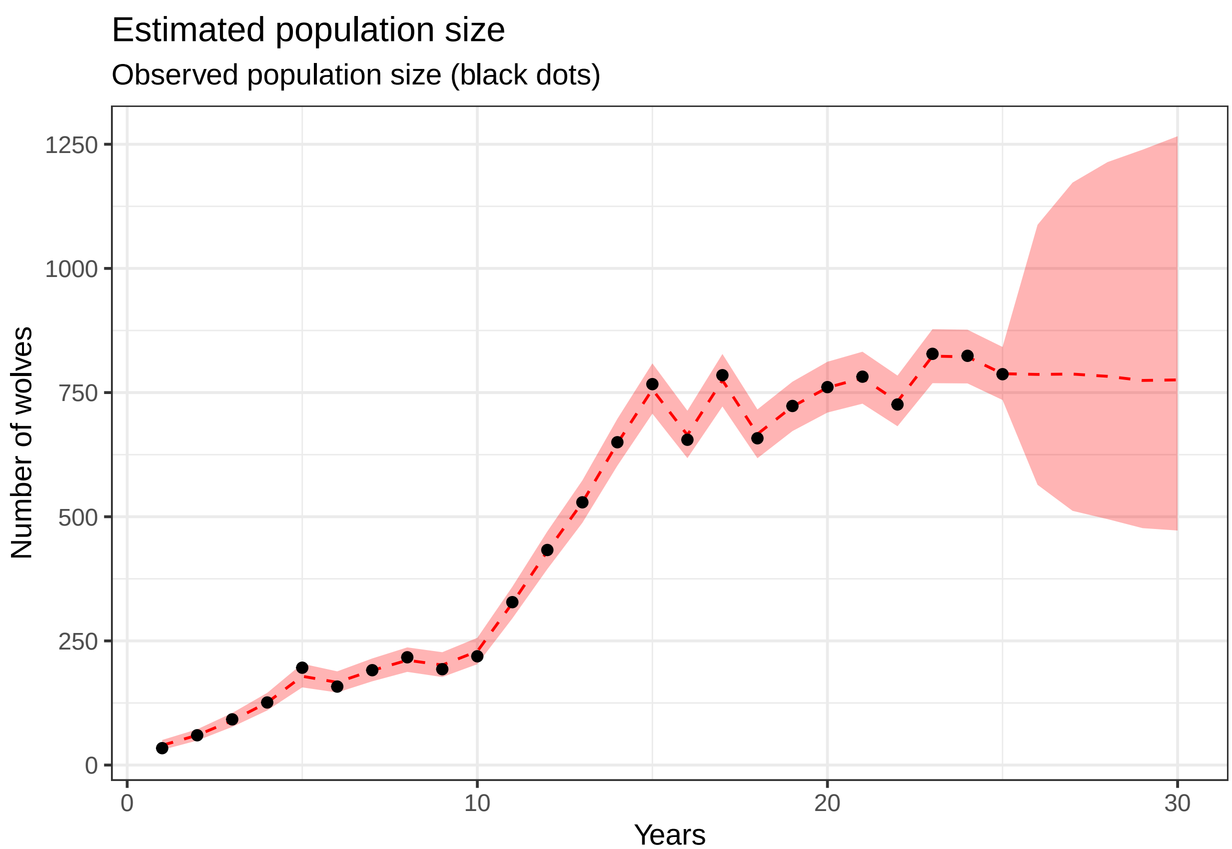



```
tempH
```

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

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
  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
```

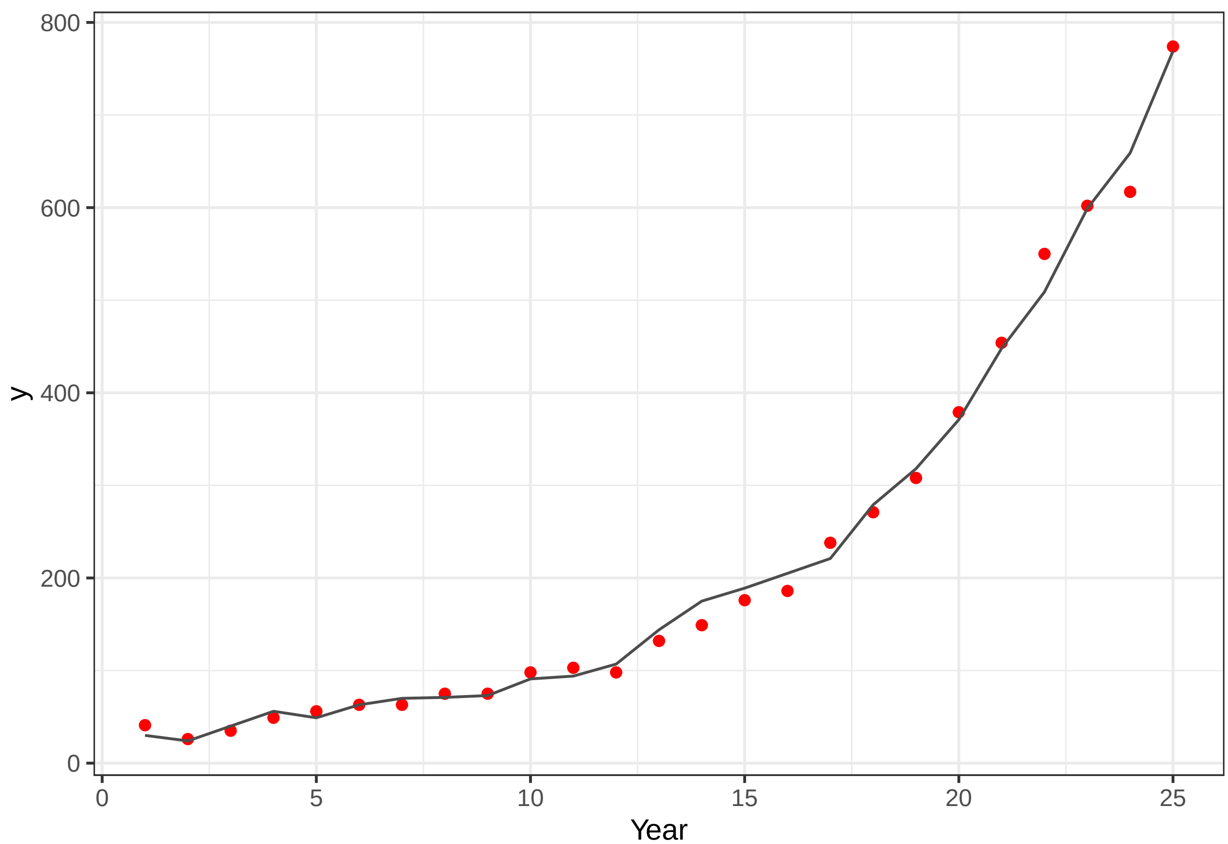
```r
    for (t in (nyears - 5):(nyears - 1)) {
      u = ssm_sim5$N[t] * (lambda - H)
      ssm_sim5$N[t + 1] = rpois(1, u)
    }
  }

  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 = 50000
  )

  #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)
}
```

```
## [1] 5
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
```

```
##    Unobserved stochastic nodes: 12
##    Total graph size: 55
##
## Initializing model
##
## [1] 1.216568
## [1] 0.1
## [1] 10
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 17
##    Total graph size: 80
##
## Initializing model
##
## [1] 1.230923
## [1] 0.1
## [1] 15
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 22
##    Total graph size: 105
##
## Initializing model
##
## [1] 1.244047
## [1] 0.1
## [1] 20
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 27
##    Total graph size: 130
##
## Initializing model
##
## [1] 1.240956
## [1] 0.1
## [1] 25
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 32
##    Total graph size: 155
```

```
##
## Initializing model
##
## [1] 1.243466
## [1] 0.1
```

```
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 0.1 0.1 0.1 0.1
```

```
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))%>%
```

```r
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")
```