Reproduire les résultats de Harvest models of small populations of a large carnivore using Bayesian forecasting par Andrén et al. 2020

Olivier Gimenez

02/09/2020, 12/01/2023

Motivation

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

Les données sont disponibles, mais pas le code. Haha, Erlend le dernier auteur est un fervent défenseur de la science reproductible, c'est loupé sur ce coup-là. Je suppose que les analyses ont été faites par Hobbs, qui a fait plusieurs papiers avec un modèle approchant. Voir par exemple :

- Raiho AM, Hooten MB, Bates S, Hobbs NT (2015) Forecasting the Effects of Fertility Control on Overabundant Ungulates: White-Tailed Deer in the National Capital Region. PLoS ONE 10(12): e0143122. doi:10.1371/journal.pone.0143122
- Hobbs, N.T., Andrén, H., Persson, J., Aronsson, M. and Chapron, G. (2012), Native predators reduce harvest of reindeer by Sámi pastoralists. Ecological Applications, 22: 1640-1654. doi:10.1890/11-1309.1
- Ketz, A. C., T. L. Johnson, R. J. Monello, and N. T. Hobbs. 2016. Informing management with monitoring data: the value of Bayesian forecasting. Ecosphere 7(11):e01587. <10.1002/ecs2.1587>

Données

On récupère les données de monitoring et harvest pour le lynx. Les colonnes sont : * year – the year of census (February) * run – the run in the data * country – code for country; S = Sweden and N = Norway * region – code for management region; Z = Jämtland, Y = Västernorrland, AC = Västerbotten, BD = Norrbotten, 2 – 8 = the different large carnivore management regions in Norway (2 – 8) * census – number of lynx family groups censused in that year in that region * harvest – total number of lynx harvested in that year in that region * harvest_F_>1yr – number of females older than one year harvested in that year in that region * harvest_F_kitten – number of female kittens (10 months old) harvested in that year in that region

```r
dat <- read.csv("eap2063-sup-0003-datas1.csv")
dat
```

```
##     year run country region census harvest harvest_F_.1yr harvest_F_kitten
## 1   1998   1       S      Z     82      44             15                1
## 2   1999   2       S      Z     84      30             14                2
## 3   2000   3       S      Z     63      52             14                7
## 4   2001   4       S      Z     49      39             12                7
## 5   2002   5       S      Z     44      31              8                5
```

```
## 6   2003   6     S   Z    39   19      3      1
## 7   2004   7     S   Z    32   17      4      2
## 8   2005   8     S   Z    42    8      2      0
## 9   2006   9     S   Z    44   16      6      3
## 10  2007  10     S   Z    42   16      5      3
## 11  2008  11     S   Z    53   26      7      6
## 12  2009  12     S   Z    53   55     22      7
## 13  2010  13     S   Z    35   42     15      2
## 14  2011  14     S   Z    39   59     24      6
## 15  2012  15     S   Z    31   18      5      3
## 16  2013  16     S   Z    14    9      3      1
## 17  2014  17     S   Z    20    1      1      0
## 18  2015  18     S   Z    33    8      2      2
## 19  2016  19     S   Z    38   38      5     15
## 20  2017  20     S   Z    36   36      9      4
## 21  1998   1     S   Y    41   13      4      1
## 22  1999   2     S   Y    37   16      6      2
## 23  2000   3     S   Y    30   13      5      3
## 24  2001   4     S   Y    28    8      2      2
## 25  2002   5     S   Y    20    5      3      0
## 26  2003   6     S   Y    19    6      3      0
## 27  2004   7     S   Y     7    1      0      0
## 28  2005   8     S   Y    14    0      0      0
## 29  2006   9     S   Y    11    2      0      0
## 30  2007  10     S   Y    12    0      0      0
## 31  2008  11     S   Y    16    0      0      0
## 32  2009  12     S   Y    17    4      2      0
## 33  2010  13     S   Y    18    8      3      1
## 34  2011  14     S   Y    24   12      2      1
## 35  2012  15     S   Y    26    7      2      0
## 36  2013  16     S   Y    14    8      2      0
## 37  2014  17     S   Y    16    6      3      1
## 38  2015  18     S   Y    16    2      0      0
## 39  2016  19     S   Y    18   12      3      2
## 40  2017  20     S   Y    19    9      2      0
## 41  1998   1     S   AC   36    5      1      0
## 42  1999   2     S   AC   36    7      2      0
## 43  2000   3     S   AC   34   18      6      1
## 44  2001   4     S   AC   38   15      8      2
## 45  2002   5     S   AC   29   16      8      2
## 46  2003   6     S   AC   24    7      3      0
## 47  2004   7     S   AC   21    8      3      0
## 48  2005   8     S   AC   31    4      0      0
## 49  2006   9     S   AC   31    2      0      0
## 50  2007  10     S   AC   23    6      2      0
## 51  2008  11     S   AC   37    7      2      0
## 52  2009  12     S   AC   41   23      8      1
## 53  2010  13     S   AC   28   13      6      1
## 54  2011  14     S   AC   32   11      4      1
## 55  2012  15     S   AC   34   26      6      4
## 56  2013  16     S   AC   22   34     10      2
## 57  2014  17     S   AC   13    7      2      0
## 58  2015  18     S   AC   13    3      1      1
## 59  2016  19     S   AC   27   12      4      1
```

```
## 60  2017  20        S     AC      28       15              7              0
## 61  1998   1        S     BD      35        3              1              0
## 62  1999   2        S     BD      37        4              1              0
## 63  2000   3        S     BD      23        1              1              0
## 64  2001   4        S     BD      34        1              0              0
## 65  2002   5        S     BD      39        0              0              0
## 66  2003   6        S     BD      32        0              0              0
## 67  2004   7        S     BD      25        0              0              0
## 68  2005   8        S     BD      32        2              1              0
## 69  2006   9        S     BD      23        1              0              1
## 70  2007  10        S     BD      24        2              0              0
## 71  2008  11        S     BD      33        0              0              0
## 72  2009  12        S     BD      37        2              0              0
## 73  2010  13        S     BD      33       19              8              2
## 74  2011  14        S     BD      44       13              5              1
## 75  2012  15        S     BD      43       28             13              4
## 76  2013  16        S     BD      30       25              7              4
## 77  2014  17        S     BD      17        8              3              1
## 78  2015  18        S     BD      32        8              2              2
## 79  2016  19        S     BD      28       14              3              2
## 80  2017  20        S     BD      26       23              6              3
## 81  1996   1        N      2      14       15              1              2
## 82  1997   2        N      2      20       18              4              2
## 83  1998   3        N      2      14       29              8              4
## 84  1999   4        N      2      20       21              7              2
## 85  2000   5        N      2      12       18              5              3
## 86  2001   6        N      2      13       16              7              0
## 87  2002   7        N      2       9       14              6              2
## 88  2003   8        N      2       4       15              4              2
## 89  2004   9        N      2       7        7              2              1
## 90  2005  10        N      2      13       10              3              2
## 91  2006  11        N      2      13        6              2              2
## 92  2007  12        N      2      13       10              4              1
## 93  2008  13        N      2      14       22              4              2
## 94  2009  14        N      2      19       27              8              2
## 95  2010  15        N      2      17       28             10              3
## 96  2011  16        N      2      14       26              9              2
## 97  2012  17        N      2      16       16              4              2
## 98  2013  18        N      2      16       23              9              2
## 99  2014  19        N      2      16       29             11              5
## 100 2015  20        N      2      16       37              9              3
## 101 2016  21        N      2       9       21              8              0
## 102 2017  22        N      2       9        5              3              0
## 103 1996   1        N      3       1        4              0              1
## 104 1997   2        N      3       3        5              0              0
## 105 1998   3        N      3       2       11              4              0
## 106 1999   4        N      3       3       14              3              1
## 107 2000   5        N      3       5        9              2              2
## 108 2001   6        N      3       5       10              6              1
## 109 2002   7        N      3       7       12              5              2
## 110 2003   8        N      3       3        5              3              0
## 111 2004   9        N      3       3        1              1              0
## 112 2005  10        N      3       6        1              1              0
## 113 2006  11        N      3       5        3              2              0
```

```
## 114 2007  12         N       3       6       6               4                 1
## 115 2008  13         N       3       5      11               4                 2
## 116 2009  14         N       3       6      10               2                 3
## 117 2010  15         N       3       4       9               5                 0
## 118 2011  16         N       3       4      11               3                 0
## 119 2012  17         N       3       5       5               0                 1
## 120 2013  18         N       3       7       8               2                 0
## 121 2014  19         N       3       5      11               2                 1
## 122 2015  20         N       3       7       9               1                 0
## 123 2016  21         N       3       3       6               3                 0
## 124 2017  22         N       3       5       4               1                 0
## 125 1996   1         N       4       2       0               0                 0
## 126 1997   2         N       4       3       0               0                 0
## 127 1998   3         N       4       6       0               0                 0
## 128 1999   4         N       4       6      10               2                 2
## 129 2000   5         N       4       1      11               2                 1
## 130 2001   6         N       4       5       7               2                 2
## 131 2002   7         N       4       5      11               6                 2
## 132 2003   8         N       4       5       5               1                 0
## 133 2004   9         N       4       6       7               3                 0
## 134 2005  10         N       4       7       4               3                 1
## 135 2006  11         N       4       6       6               3                 1
## 136 2007  12         N       4       6       5               2                 0
## 137 2008  13         N       4       5       7               0                 2
## 138 2009  14         N       4       7       6               2                 0
## 139 2010  15         N       4       9       6               2                 0
## 140 2011  16         N       4       6      11               4                 0
## 141 2012  17         N       4       5       6               1                 0
## 142 2013  18         N       4       1       3               1                 0
## 143 2014  19         N       4       5       0               0                 0
## 144 2015  20         N       4       4       2               0                 0
## 145 2016  21         N       4       1       0               0                 0
## 146 2017  22         N       4       3       0               0                 0
## 147 1996   1         N       5       9      10               2                 1
## 148 1997   2         N       5       7       9               3                 0
## 149 1998   3         N       5      11      14               6                 1
## 150 1999   4         N       5      11      15               3                 2
## 151 2000   5         N       5       6      12               5                 1
## 152 2001   6         N       5       9      12               5                 2
## 153 2002   7         N       5       8      14               6                 3
## 154 2003   8         N       5       7      17               8                 4
## 155 2004   9         N       5       8       9               0                 1
## 156 2005  10         N       5       7      12               4                 0
## 157 2006  11         N       5      10       7               3                 0
## 158 2007  12         N       5      11       9               3                 1
## 159 2008  13         N       5      10      11               5                 0
## 160 2009  14         N       5       9      14               3                 1
## 161 2010  15         N       5       9      10               3                 0
## 162 2011  16         N       5      11       9               3                 1
## 163 2012  17         N       5       6       8               5                 0
## 164 2013  18         N       5       5       5               1                 0
## 165 2014  19         N       5       4       0               0                 0
## 166 2015  20         N       5       2       0               0                 0
## 167 2016  21         N       5       7       0               0                 0
```

```
## 168 2017  22        N       5       9       0              0                0
## 169 1996   1        N       6      20      34             10                4
## 170 1997   2        N       6      26      40             10                2
## 171 1998   3        N       6      14      32             12                1
## 172 1999   4        N       6      14      14              4                3
## 173 2000   5        N       6      14      15              2                2
## 174 2001   6        N       6       9       7              3                2
## 175 2002   7        N       6      11      17              3                4
## 176 2003   8        N       6      11       9              2                1
## 177 2004   9        N       6      14       3              2                0
## 178 2005  10        N       6      14      14              4                4
## 179 2006  11        N       6      17      18              6                2
## 180 2007  12        N       6      15      29              6                4
## 181 2008  13        N       6      23      30              9                4
## 182 2009  14        N       6      26      36             16                8
## 183 2010  15        N       6      20      59             22                3
## 184 2011  16        N       6      18      52             16                4
## 185 2012  17        N       6      14      17              7                3
## 186 2013  18        N       6       8      15              6                0
## 187 2014  19        N       6      12       7              2                1
## 188 2015  20        N       6      17      18              6                0
## 189 2016  21        N       6      14      31             11                2
## 190 2017  22        N       6      19      33             11                5
## 191 1996   1        N       7      12      14              6                2
## 192 1997   2        N       7      14      16              4                3
## 193 1998   3        N       7      10      16              6                1
## 194 1999   4        N       7      16      11              5                0
## 195 2000   5        N       7      15      20              6                5
## 196 2001   6        N       7       5      16              6                2
## 197 2002   7        N       7       6      13              6                0
## 198 2003   8        N       7       5       7              4                0
## 199 2004   9        N       7       2       5              2                0
## 200 2005  10        N       7       4       2              1                0
## 201 2006  11        N       7       6       0              0                0
## 202 2007  12        N       7       8       0              0                0
## 203 2008  13        N       7       9       4              0                1
## 204 2009  14        N       7      14       8              4                1
## 205 2010  15        N       7       6      16              8                1
## 206 2011  16        N       7       8      12              4                1
## 207 2012  17        N       7       8       9              2                1
## 208 2013  18        N       7      10       6              4                0
## 209 2014  19        N       7       4      13              4                0
## 210 2015  20        N       7       5       4              2                0
## 211 2016  21        N       7       6       5              1                0
## 212 2017  22        N       7       6       6              1                0
## 213 1996   1        N       8       5       4              2                0
## 214 1997   2        N       8       7       5              1                1
## 215 1998   3        N       8       7      11              5                0
## 216 1999   4        N       8       5       1              1                0
## 217 2000   5        N       8       6      10              6                2
## 218 2001   6        N       8       6      13              7                1
## 219 2002   7        N       8       8      11              2                0
## 220 2003   8        N       8      10       4              1                0
## 221 2004   9        N       8       3       3              2                0
```

```
## 222 2005   10        N       8       3       1              0                0
## 223 2006   11        N       8       5       0              0                0
## 224 2007   12        N       8      12       1              1                0
## 225 2008   13        N       8       9       4              1                0
## 226 2009   14        N       8       9       8              4                0
## 227 2010   15        N       8      15       7              1                0
## 228 2011   16        N       8      11      16              7                1
## 229 2012   17        N       8      13      18              5                4
## 230 2013   18        N       8      10      13              3                2
## 231 2014   19        N       8       5      10              4                0
## 232 2015   20        N       8       8       5              1                2
## 233 2016   21        N       8       9       3              2                0
## 234 2017   22        N       8       6       4              2                1
```

```r
dat %>%
  count(region)
```

```
##    region  n
## 1       2 22
## 2       3 22
## 3       4 22
## 4       5 22
## 5       6 22
## 6       7 22
## 7       8 22
## 8      AC 20
## 9      BD 20
## 10      Y 20
## 11      Z 20
```

```r
dat %>%
  count(country)
```

```
##   country   n
## 1       N 154
## 2       S  80
```

Le modèle

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

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

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

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

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

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

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

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

```
lynx_model <- function(){

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

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

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

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

}
```

Dans le papier, Henrik fait des regroupements d'aires de gestion, et applique le modèle à chacun de ces regroupements.

Northern Sweden: management areas Z, Y, BD and AC

On regroupe.

```
dat %>%
  filter(region == "Z" | region == "Y" | region == "BD" | region == "AC") %>%
  select(year, census, harvest) %>%
  group_by(year) %>%
  summarize(census = sum(census),
            harvest = sum(harvest)) -> dat1
```

On prépare les données.

```
bugs.data <- list(
    nyears = 20,
    y = dat1$census,
    harvest = dat1$harvest)
```

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

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

Allez zooh, on lance la machine!

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

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 22
##    Total graph size: 147
##
## Initializing model
```

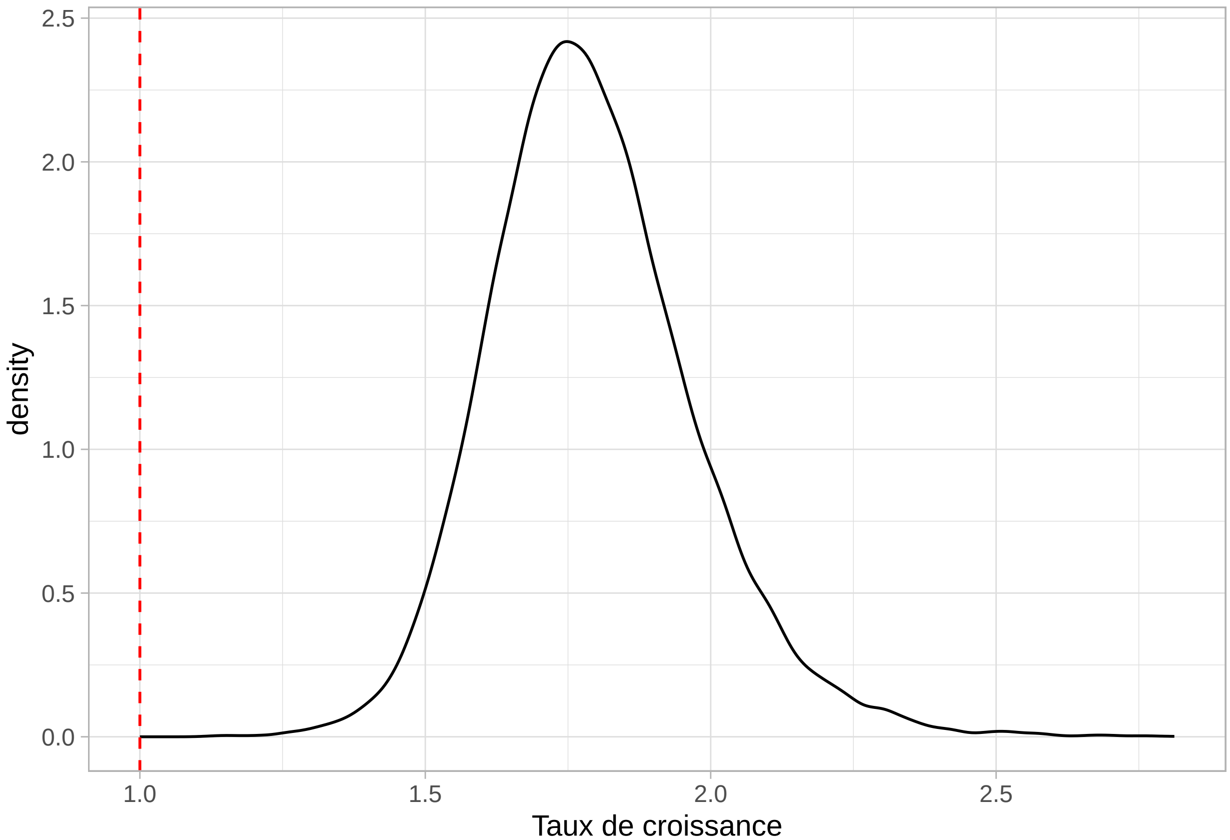
Jetons un coup d'oeil aux estimations.

```
res <- print(lynx_mod, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a
##  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]      192.129  13.269 167.237 191.683 219.178 1.001 13000
## N[2]      191.033  13.362 165.975 190.692 218.107 1.001 15000
## N[3]      155.205  11.209 133.897 154.988 178.083 1.001 15000
## N[4]      146.640  11.234 125.618 146.257 169.336 1.001 15000
## N[5]      130.316  10.583 110.360 129.947 152.121 1.001 12000
## N[6]      111.188   9.825  92.631 110.814 131.280 1.001 15000
## N[7]       88.442   8.596  72.352  88.089 106.074 1.001 15000
## N[8]      114.857  10.234  95.812 114.451 135.873 1.001 15000
## N[9]      108.788   9.771  90.737 108.383 129.045 1.001 12000
## N[10]     103.668   9.489  85.917 103.402 123.164 1.001 15000
## N[11]     137.313  11.039 116.583 136.954 159.674 1.001 15000
## N[12]     150.709  10.887 130.301 150.381 172.839 1.001 15000
```

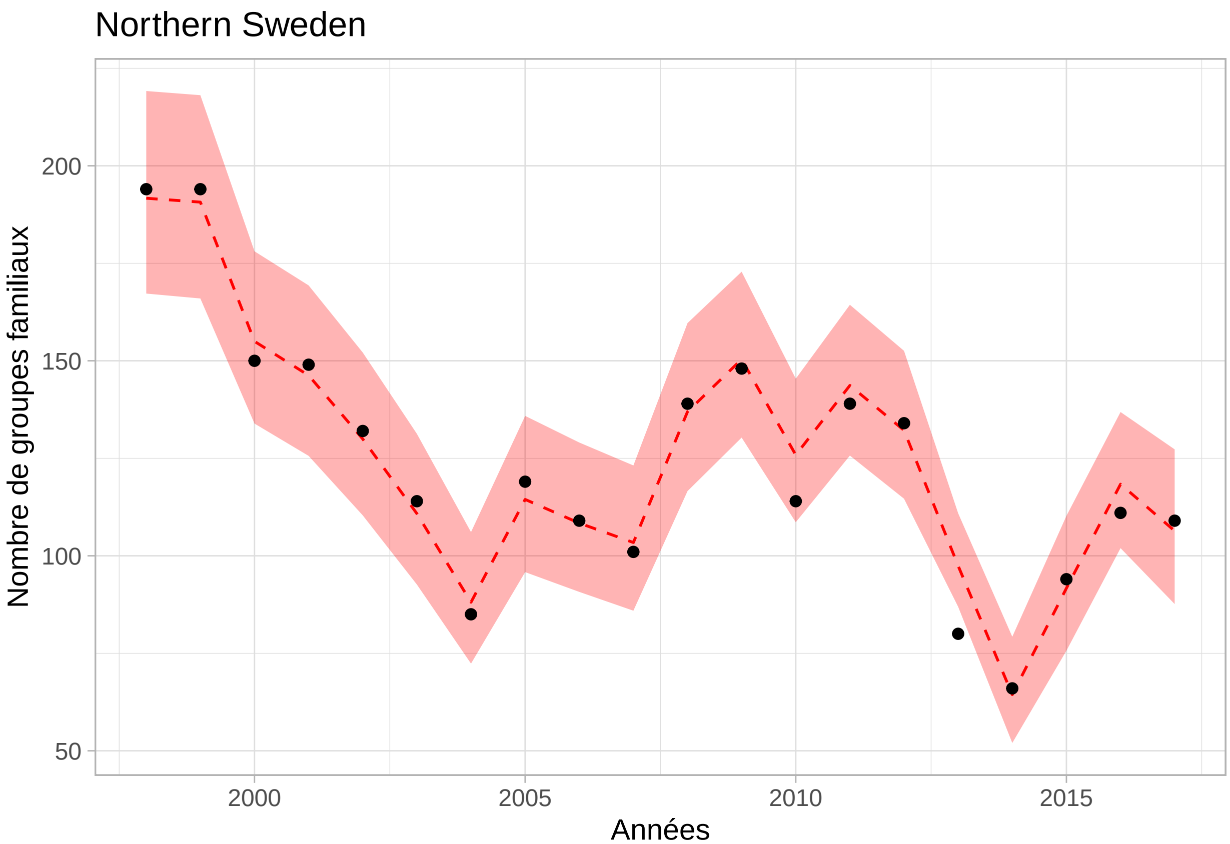
```
## N[13]     126.194   9.377 108.600 125.854 145.414 1.001 15000
## N[14]     143.977   9.813 125.692 143.686 164.363 1.001 15000
## N[15]     132.483   9.682 114.630 132.173 152.557 1.001 14000
## N[16]      97.786   6.144  86.925  97.418 110.906 1.001  9500
## N[17]      64.701   7.025  51.998  64.394  79.246 1.001  8500
## N[18]      92.054   8.826  75.652  91.683 110.233 1.001 15000
## N[19]     118.606   8.885 101.958 118.366 136.867 1.001 15000
## N[20]     106.707  10.164  87.619 106.378 127.290 1.001 15000
## lambda      1.793   0.181   1.481   1.779   2.194 1.001 12000
## sigmaProc   0.407   0.096   0.259   0.394   0.633 1.001 11000
## tauProc     7.040   3.207   2.497   6.450  14.949 1.001 11000
## deviance  154.753   6.457 143.915 154.138 169.144 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 = 20.9 and DIC = 175.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
#  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
#  filter(str_detect(parameter, "lambda")) %>%
  ggplot() +
  aes(x = lambda) +
  geom_density() +
  geom_vline(xintercept = 1, lty = "dashed", color = "red") +
  labs(x = "Taux de croissance")
```



Ensuite les projections.

```
northern_sweden <- lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
  filter(str_detect(parameter, "N")) %>%
  group_by(parameter) %>%
  summarize(medianN = median(value),
            lci = quantile(value, probs = 2.5/100),
            uci = quantile(value, probs = 97.5/100)) %>%
  mutate(an = parse_number(parameter) + 1997) %>%
  arrange(an) %>%
  ggplot() +
  geom_ribbon(aes(x = an, y = medianN, ymin = lci, ymax = uci), fill = "red", alpha = 0.3) +
  geom_line(aes(x = an, y = medianN), lty = "dashed", color = "red") +
#  geom_point(aes(x = an, y = medianN), color = "red") +
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1997 + 1:unique(nyears), y = y)) +
  labs(y = "Nombre de groupes familiaux",
       x = "Années",
       title = "Northern Sweden")
northern_sweden
```



Northern Norway: management areas 6, 7, 8

Idem qu'au-dessus.

```
dat %>%
  filter(region == "6" | region == "7" | region == "8") %>%
  select(year, census, harvest) %>%
  group_by(year) %>%
  summarize(census = sum(census),
            harvest = sum(harvest)) -> dat1
```

```
bugs.data <- list(
    nyears = 22,
    y = dat1$census,
    harvest = dat1$harvest)
```

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

```
bugs.monitor <- c("lambda", "sigmaProc","N", "tauProc")
bugs.chains <- 2
bugs.inits <- function(){
    list(
    )
}
```

Allez zooh, on lance la machine!

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

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 24
##    Total graph size: 161
##
## Initializing model
```

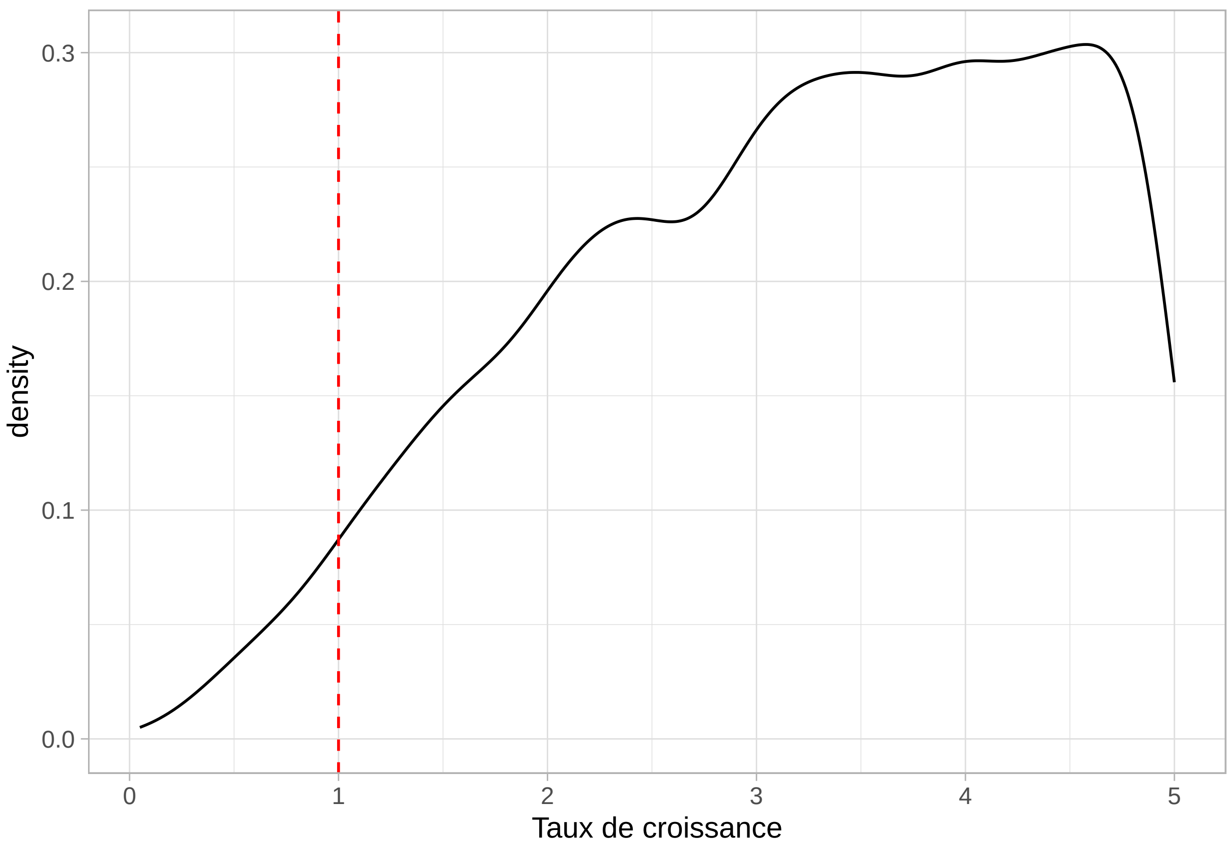
Jetons un coup d'oeil aux estimations.

```r
res <- print(lynx_mod, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a
##  2 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 10000 iterations saved
##           mu.vect sd.vect   2.5%    50%   97.5%  Rhat n.eff
## N[1]       37.210   6.206  26.153  36.785  50.936 1.001 10000
## N[2]       46.790   7.193  34.032  46.319  62.761 1.001 10000
## N[3]       30.481   5.525  20.720  30.121  42.133 1.001 10000
## N[4]       35.240   5.505  25.663  34.869  47.167 1.001  6000
## N[5]       35.379   6.244  24.464  34.846  49.143 1.001 10000
## N[6]       19.757   4.448  12.088  19.408  29.405 1.001 10000
## N[7]       24.692   4.961  15.897  24.383  35.343 1.001 10000
## N[8]       26.180   4.759  17.497  25.813  36.425 1.001  9400
## N[9]       19.099   4.183  12.101  18.738  28.317 1.001  7900
## N[10]      21.886   4.406  13.730  21.568  31.288 1.001 10000
## N[11]      28.246   5.134  19.604  27.788  39.537 1.001 10000
## N[12]      36.359   5.533  25.765  35.997  48.051 1.001 10000
## N[13]      42.944   6.086  30.729  42.809  55.326 1.001  9600
## N[14]      50.924   7.529  36.570  51.319  65.340 1.001 10000
## N[15]      40.714   6.368  29.004  40.441  54.009 1.001 10000
## N[16]      36.545   5.995  25.653  36.195  48.962 1.001 10000
## N[17]      35.091   6.336  23.959  34.523  49.045 1.001 10000
## N[18]      28.349   5.777  18.261  27.863  40.416 1.001  4400
## N[19]      20.859   4.843  12.654  20.379  32.143 1.001 10000
## N[20]      30.924   5.267  20.763  30.751  41.769 1.001 10000
## N[21]      29.181   5.750  19.306  28.664  42.134 1.001 10000
## N[22]      30.612   5.459  20.925  30.281  42.181 1.001 10000
## lambda      3.201   1.161   0.846   3.327   4.923 1.001 10000
## sigmaProc   2.940   0.505   2.128   2.876   4.101 1.001 10000
## tauProc     0.126   0.041   0.059   0.121   0.221 1.001 10000
```

```
## deviance  138.185   6.721 126.810 137.502 153.000 1.002  1700
##
## 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 = 22.6 and DIC = 160.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```

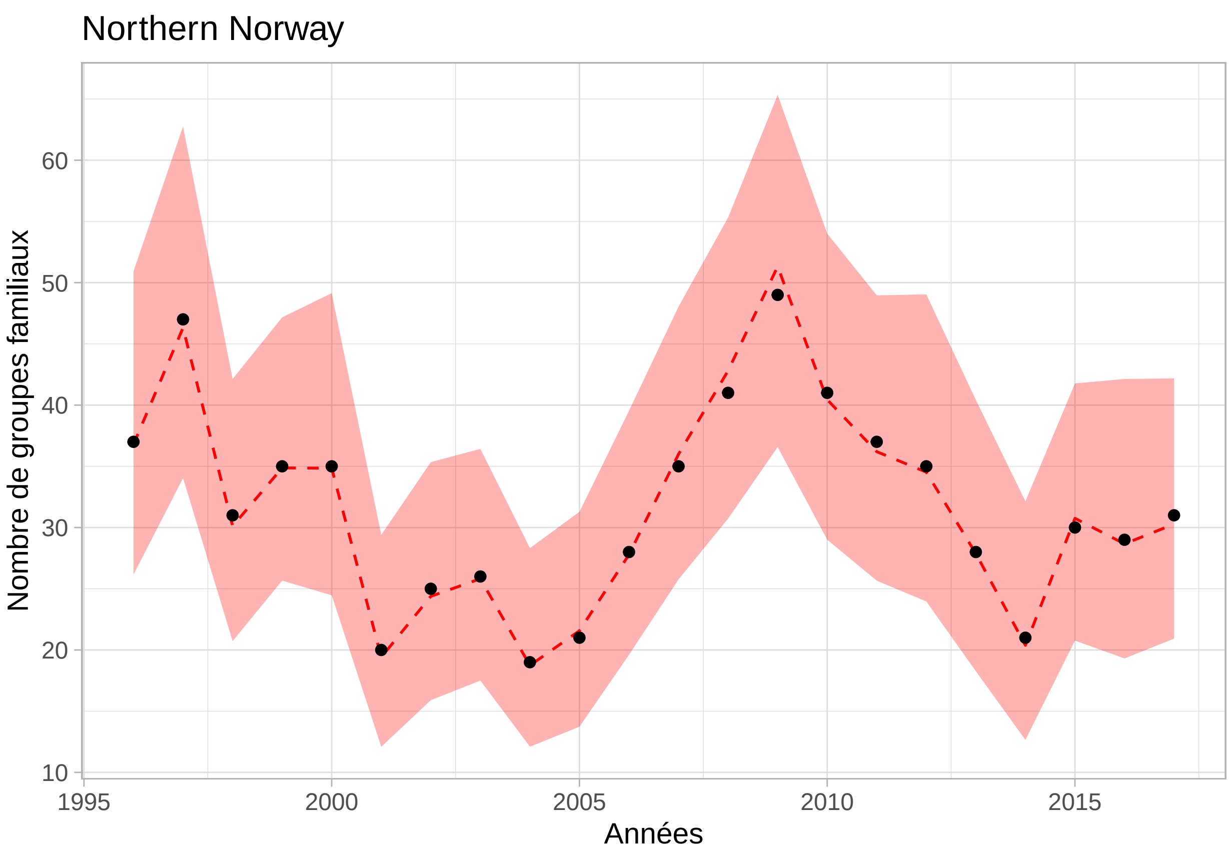
```
lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
#  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
#  filter(str_detect(parameter, "lambda")) %>%
  ggplot() +
  aes(x = lambda) +
  geom_density() +
  geom_vline(xintercept = 1, lty = "dashed", color = "red") +
  labs(x = "Taux de croissance")
```



Ensuite les projections.

```
northern_norway <- lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
  filter(str_detect(parameter, "N")) %>%
  group_by(parameter) %>%
  summarize(medianN = median(value),
```

```
            lci = quantile(value, probs = 2.5/100),
            uci = quantile(value, probs = 97.5/100)) %>%
  mutate(an = parse_number(parameter) + 1995) %>%
  arrange(an) %>%
  ggplot() +
  geom_ribbon(aes(x = an, y = medianN, ymin = lci, ymax = uci), fill = "red", alpha = 0.3) +
  geom_line(aes(x = an, y = medianN), lty = "dashed", color = "red") +
#  geom_point(aes(x = an, y = medianN), color = "red") +
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  labs(y = "Nombre de groupes familiaux",
       x = "Années",
       title = "Northern Norway")
northern_norway
```



Southern Norway: management areas 2, 3, 4 and 5

On applique le modèle exponentiel au dernier regroupement.

```
dat %>%
  filter(region == "2" | region == "3" | region == "4" | region == "5") %>%
  select(year, census, harvest) %>%
  group_by(year) %>%
  summarize(census = sum(census),
            harvest = sum(harvest)) -> dat1
```

```
bugs.data <- list(
    nyears = 22,
    y = dat1$census,
    harvest = dat1$harvest)
```

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

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

Allez zooh, on lance la machine!

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

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 22
##    Unobserved stochastic nodes: 24
##    Total graph size: 161
##
## Initializing model
```

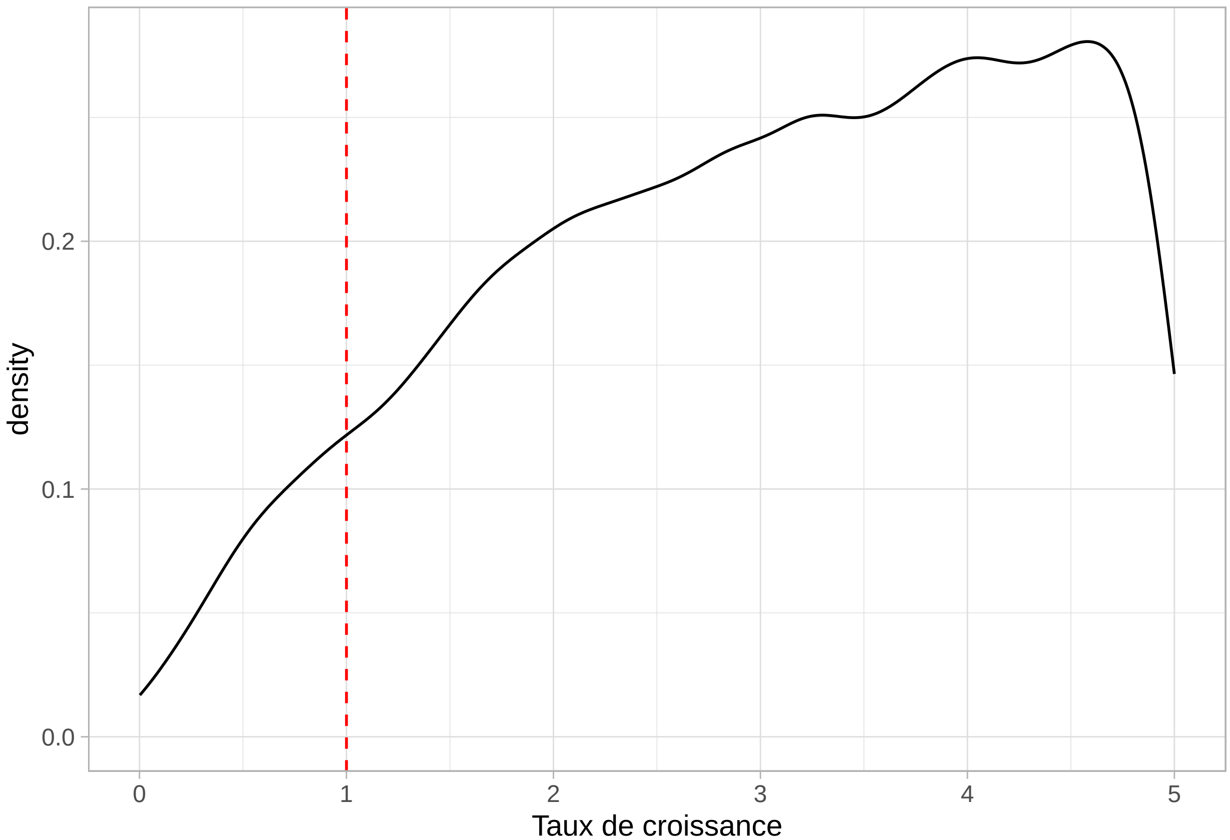
Jetons un coup d'oeil aux estimations.

```
res <- print(lynx_mod, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a7
##  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]      26.996   5.523 17.355 26.705  38.067 1.001 15000
## N[2]      34.097   5.776 23.254 34.215  45.899 1.001 15000
## N[3]      32.795   5.698 22.681 32.446  44.827 1.001  5700
## N[4]      39.656   6.364 28.207 39.342  53.155 1.001 15000
## N[5]      23.624   4.866 14.984 23.278  33.989 1.001 15000
## N[6]      31.718   5.700 21.655 31.337  44.457 1.001 15000
## N[7]      28.719   5.385 19.131 28.407  40.166 1.001  3300
## N[8]      18.777   4.325 11.274 18.455  28.280 1.001 15000
```

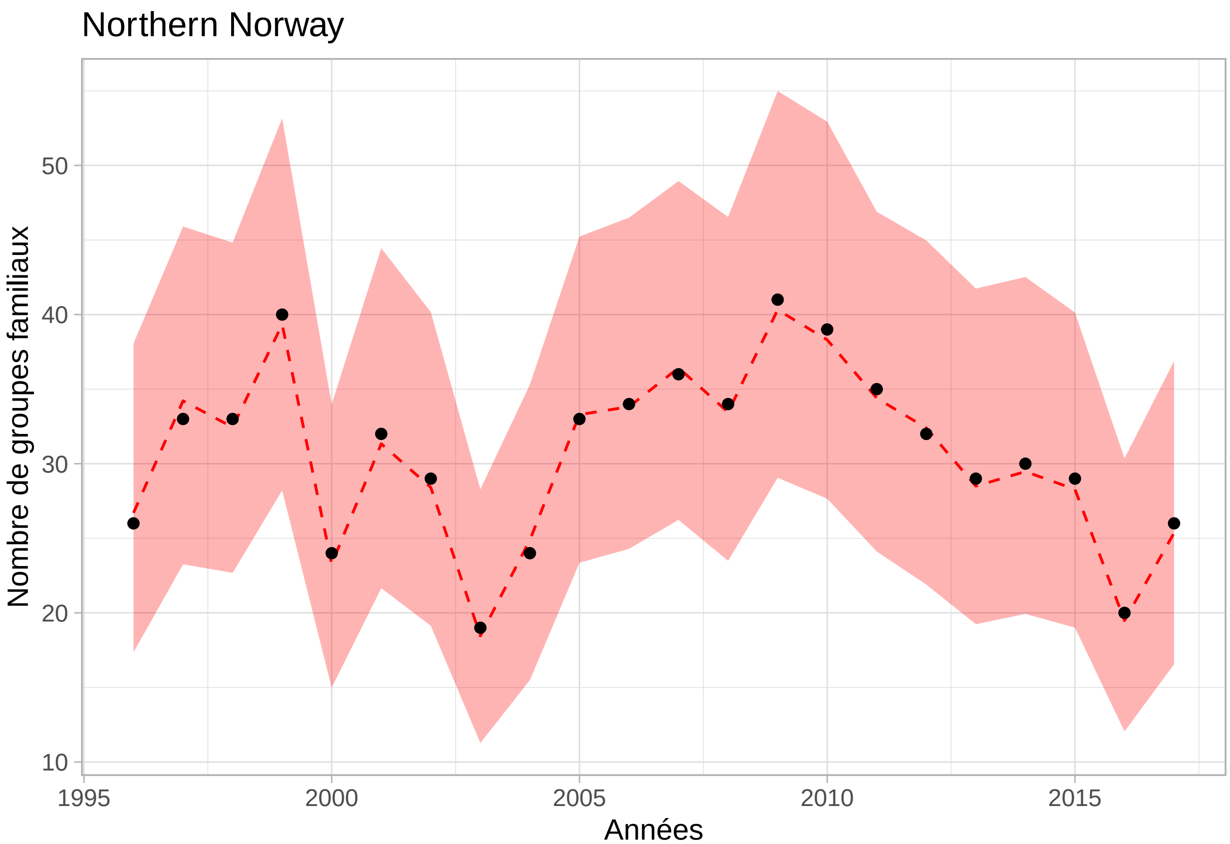
```
## N[9]       24.804   5.086  15.494  24.889  35.309 1.001 15000
## N[10]      33.631   5.460  23.361  33.292  45.233 1.001 15000
## N[11]      34.182   5.712  24.293  33.819  46.498 1.001 15000
## N[12]      36.860   5.712  26.233  36.425  48.951 1.001 15000
## N[13]      33.845   5.875  23.490  33.452  46.542 1.001 15000
## N[14]      40.692   6.556  29.059  40.318  54.990 1.001 15000
## N[15]      38.739   6.405  27.652  38.305  52.933 1.001 15000
## N[16]      34.666   5.887  24.114  34.397  46.896 1.001 15000
## N[17]      32.721   6.063  21.894  32.394  44.970 1.001 15000
## N[18]      28.984   5.665  19.237  28.497  41.745 1.001 15000
## N[19]      29.886   5.669  19.930  29.475  42.519 1.001 15000
## N[20]      28.605   5.368  19.011  28.282  40.140 1.001  5300
## N[21]      19.931   4.690  12.053  19.484  30.357 1.001 15000
## N[22]      25.700   5.166  16.553  25.358  36.880 1.001 15000
## lambda      3.024   1.271   0.518   3.154   4.916 1.001 11000
## sigmaProc   3.223   0.556   2.346   3.146   4.494 1.001  5600
## tauProc     0.104   0.034   0.050   0.101   0.182 1.001  5600
## deviance  138.320   6.917 126.871 137.579 153.647 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 = 23.9 and DIC = 162.2
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```r
lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
#  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
#  filter(str_detect(parameter, "lambda")) %>%
  ggplot() +
  aes(x = lambda) +
  geom_density() +
  geom_vline(xintercept = 1, lty = "dashed", color = "red") +
  labs(x = "Taux de croissance")
```



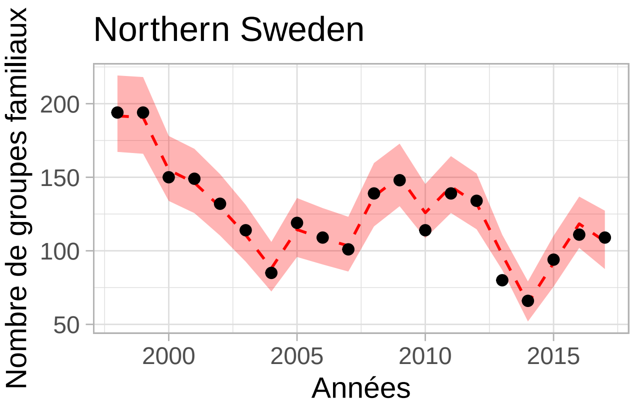
Ensuite les projections.

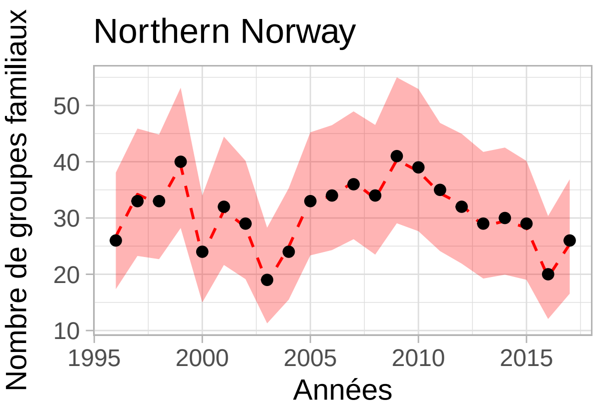
```
southern_norway <- lynx_mod$BUGSoutput$sims.matrix %>%
  as_tibble() %>%
  pivot_longer(cols = everything(),  values_to = "value", names_to = "parameter") %>%
  filter(str_detect(parameter, "N")) %>%
  group_by(parameter) %>%
  summarize(medianN = median(value),
            lci = quantile(value, probs = 2.5/100),
            uci = quantile(value, probs = 97.5/100)) %>%
  mutate(an = parse_number(parameter) + 1995) %>%
  arrange(an) %>%
  ggplot() +
  geom_ribbon(aes(x = an, y = medianN, ymin = lci, ymax = uci), fill = "red", alpha = 0.3) +
  geom_line(aes(x = an, y = medianN), lty = "dashed", color = "red") +
#  geom_point(aes(x = an, y = medianN), color = "red") +
  geom_point(data = bugs.data %>% as_tibble, aes(x = 1995 + 1:unique(nyears), y = y)) +
  labs(y = "Nombre de groupes familiaux",
       x = "Années",
       title = "Northern Norway")
southern_norway
```

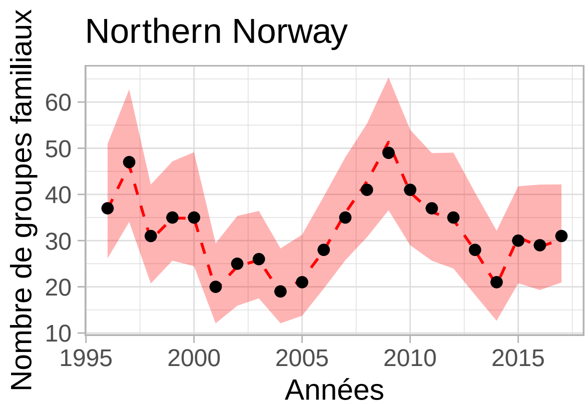


Tout ensemble - Figure 3 ou presque

```
library(patchwork)
(northern_sweden + grid::textGrob("")) / (southern_norway | northern_norway)
```







Hmm. Si l'on compare à la Figure 3 du papier, on s'aperçoit que l'ajustement du modèle exponentiel aux données est bien meilleur que celui du modèle structuré en âge développé par les auteurs. Ha!

Forecasting

Le modèle décrit l'évolution des effectifs à t en fonction des effectifs à t et permet donc de projeter les effectifs en 2018 en connaissant les effectifs de 2017 la dernière année du suivi, puis ceux de 2019 en utilisant les effectifs prédits pour 2018, et ainsi de suite. A cahque étape, il y a des erreurs qui s'accumulent. L'approche bayésienne a l'avantage de permettre de faire ces prédictions en reportant les incertitudes d'une année à l'autre. C'est ce qui fait des modèles à espace d'états en bayésien un outil très utile pour faire des projections.

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

Les auteurs proposent le modèle à espace d'états suivant :

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

et

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

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

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

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

On lit les données spécifique au modèle de décision. On a : * year – the year of census (February) * run – the run in the data * country – code for country; 1 = Sweden and 2 = Norway * census – number of lynx family groups censused in that year in that region * quota – the harvest quota for lynx based on the census result of the same year in that region * quota_1 – the harvest quota for lynx based on the census result of the year before in the region.

```r
dat <- read.csv("eap2063-sup-0004-datas2.csv")

dat %>%
  filter(country == "1") %>%
  select(year, census, quota_1) -> dat_sweden

dat_sweden
```

```
##    year census quota_1
## 1  1998    194     140
## 2  1999    194     108
## 3  2000    150      73
## 4  2001    149      72
## 5  2002    132      67
## 6  2003    114      32
## 7  2004     86      15
## 8  2005    119      28
## 9  2006    109      30
## 10 2007    102      32
## 11 2008    140      99
## 12 2009    148     127
## 13 2010    114      95
## 14 2011    139      86
## 15 2012    135      62
## 16 2013     80      24
## 17 2014     66       0
```

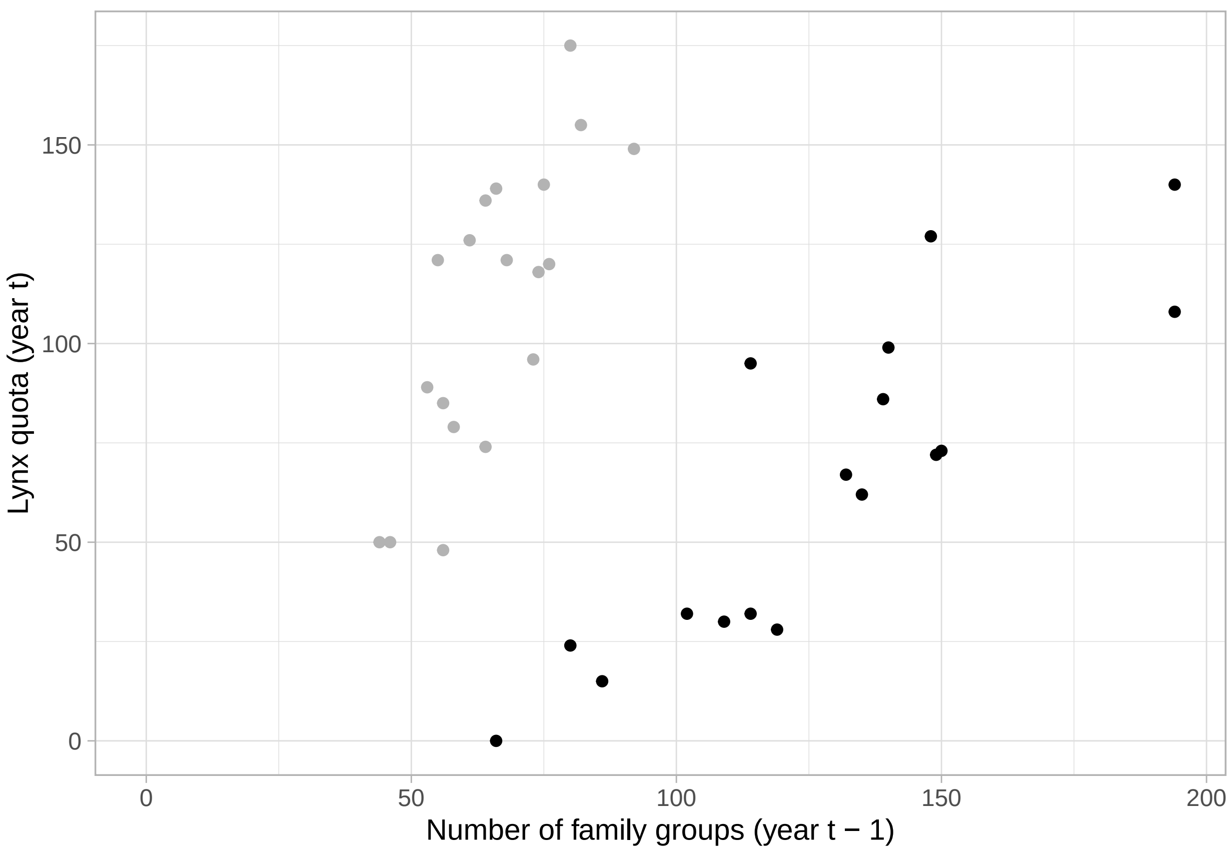
```r
dat %>%
  filter(country == "2") %>%
  select(year, census, quota_1) -> dat_norway

dat_norway
```

```
##    year census quota_1
## 1  1996     64     136
## 2  1997     82     155
## 3  1998     66     139
## 4  1999     75     140
## 5  2000     61     126
## 6  2001     55     121
## 7  2002     56      85
## 8  2003     46      50
## 9  2004     44      50
```

```
## 10 2005      56       48
## 11 2006      64       74
## 12 2007      73       96
## 13 2008      76      120
## 14 2009      92      149
## 15 2010      80      175
## 16 2011      74      118
## 17 2012      68      121
## 18 2013      58       79
## 19 2014      53       89
```

```
ggplot() +
  geom_point(data = dat_sweden, aes(x = census, y = quota_1), color = "black") +
  geom_point(data = dat_norway, aes(x = census, y = quota_1), color = "gray70") +
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of family groups (year t - 1)",
       y = "Lynx quota (year t)")
```



Modèle 1

Commençons par le modèle 1.

```
model1 <- function(){

  # Priors
```

```
  sigmaProc ~ dunif(0, 4)
  tauProc <- 1/sigmaProc^2
  b[1] ~ dnorm(0, 1/3000)

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

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

}
```

On prépare les données pour la Suède.

```
bugs.data <- list(
    nyears = 17,
    y = dat_sweden$census,
    q = dat_sweden$quota_1)
```

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

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

Allez zooh, on lance la machine!

```
mod1_sweden <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model1,
                  n.chains = bugs.chains,
                                                   n.thin = 10,
                                                   n.iter = 100000,
                                                   n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 17
##    Unobserved stochastic nodes: 19
##    Total graph size: 107
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

```
print(mod1_sweden, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##           mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## H[1]      138.777  11.758 116.657 138.472 162.740 1.001 15000
## H[2]      107.555  10.304  88.231 107.365 128.788 1.001 15000
## H[3]       72.698   8.499  57.061  72.327  90.043 1.001 15000
## H[4]       71.711   8.316  56.501  71.377  89.024 1.001 15000
## H[5]       66.531   8.079  51.535  66.236  83.179 1.001 14000
## H[6]       32.677   5.570  22.574  32.368  44.330 1.001 15000
## H[7]       16.500   3.878   9.806  16.191  24.989 1.002  3300
## H[8]       28.995   5.287  19.615  28.644  40.188 1.001  6100
## H[9]       30.799   5.425  21.134  30.488  42.261 1.001 15000
## H[10]      32.545   5.513  22.678  32.198  44.239 1.001 15000
## H[11]      97.980   9.851  79.671  97.517 118.155 1.001 15000
## H[12]     125.683  11.160 104.917 125.373 148.475 1.001 15000
## H[13]      93.699   9.568  76.114  93.389 113.380 1.001 15000
## H[14]      85.071   9.167  67.940  84.788 103.706 1.001 15000
## H[15]      61.771   7.761  47.486  61.376  77.944 1.001 15000
## H[16]      24.652   4.789  16.189  24.352  34.946 1.001 15000
## H[17]       3.832   2.026   0.853   3.508   8.577 1.001 15000
## b           0.411   0.083   0.268   0.403   0.598 1.001 15000
## sigmaProc   0.777   0.201   0.480   0.745   1.254 1.001 15000
## deviance  117.374   6.922 105.868 116.756 132.929 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 = 24.0 and DIC = 141.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Le paramètre b_1 est estimé très proche de la valeur qu'on trouve dans le Tableau 4.

```
mod1_sweden$BUGSoutput$mean$b
```

```
## [1] 0.4106349
```

Idem pour la Norvège. On prépare les données.

```
bugs.data <- list(
    nyears = 19,
    y = dat_norway$census,
    q = dat_norway$quota_1)
```

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

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

Allez zooh, on lance la machine!

```r
mod1_norway <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model1,
                  n.chains = bugs.chains,
                                                   n.thin = 10,
                                                   n.iter = 100000,
                                                   n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 21
##    Total graph size: 119
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

```r
print(mod1_norway, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a3
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##          mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## H[1]      131.950  11.069 111.444 131.564 154.696 1.001 15000
## H[2]      152.573  11.788 130.400 152.211 176.384 1.001 15000
## H[3]      135.154  10.990 114.838 134.738 157.356 1.001 15000
## H[4]      137.852  11.175 116.879 137.635 160.403 1.001  4400
## H[5]      122.531  10.627 102.837 122.258 144.231 1.001 12000
## H[6]      116.524  10.217  97.563 116.173 137.484 1.001  9500
## H[7]       85.843   8.502  69.853  85.580 103.088 1.001  4600
## H[8]       54.965   6.760  42.403  54.642  68.961 1.001 15000
## H[9]       54.412   6.670  42.149  54.111  68.179 1.001 15000
## H[10]      55.924   7.113  42.409  55.639  70.491 1.001 14000
## H[11]      78.642   8.258  63.401  78.283  95.611 1.002  2200
## H[12]      99.123   9.429  81.524  98.840 118.364 1.001 15000
## H[13]     120.284  10.467 100.861 119.937 141.500 1.001  8900
## H[14]     148.914  11.446 127.418 148.620 171.994 1.001 14000
## H[15]     170.409  12.643 146.394 170.107 196.066 1.001 15000
## H[16]     118.360  10.330  98.882 118.091 139.394 1.001  9900
```

```
## H[17]     119.762  10.340 100.780 119.347 140.926 1.001  4700
## H[18]      81.325   8.222  66.049  81.035  98.204 1.001 15000
## H[19]      88.500   8.661  72.369  88.179 106.459 1.001 15000
## b           1.611   0.105   1.413   1.607   1.827 1.001 15000
## sigmaProc   0.259   0.057   0.169   0.252   0.388 1.001 15000
## deviance  142.612   6.457 132.059 141.881 157.238 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 = 20.8 and DIC = 163.5
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Le paramètre b_1 est estimé proche de la valeur qu'on trouve dans le Tableau 4.

```
mod1_norway$BUGSoutput$mean$b
```

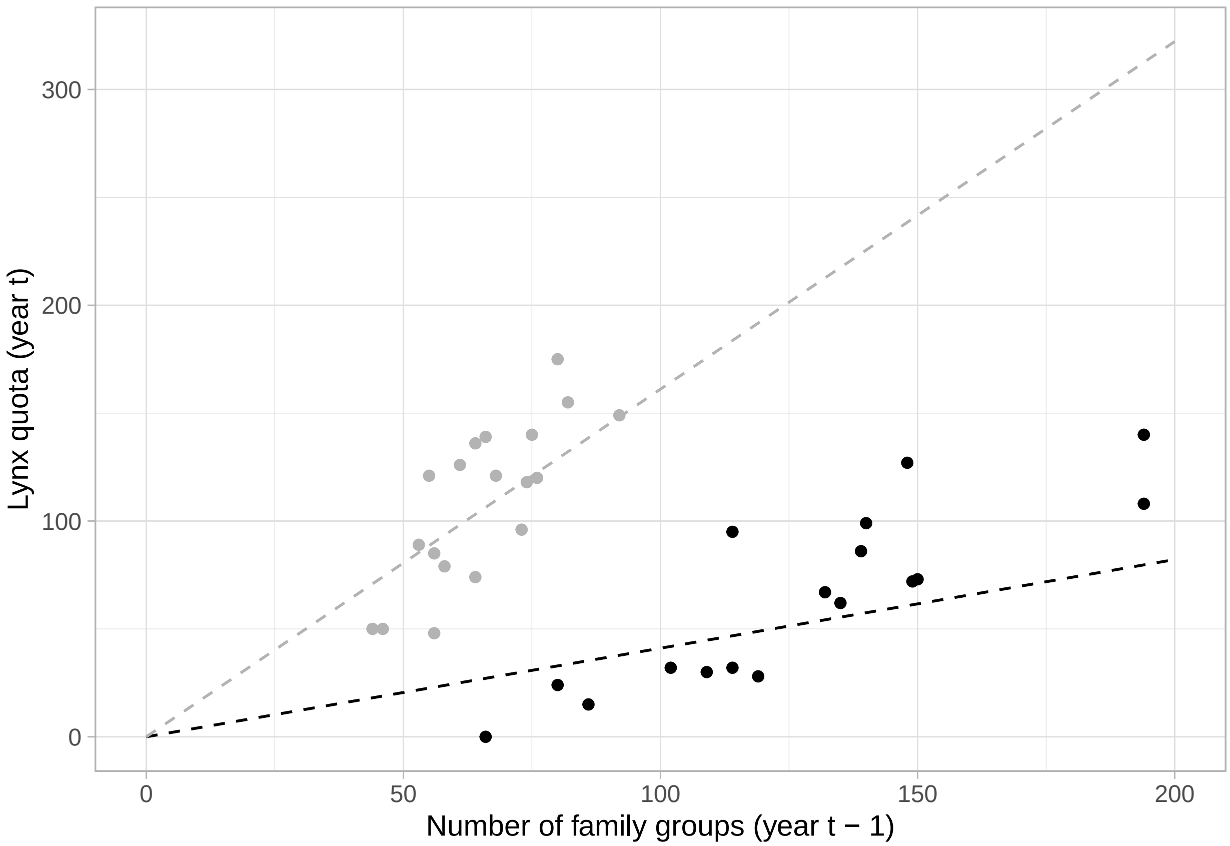
```
## [1] 1.611073
```

Graphiquement, on obtient.

```
swgrid <- seq(0, 200, length.out = length(dat_sweden$census))
nwgrid <- seq(0, 200, length.out = length(dat_norway$census))
ggplot() +
  geom_point(data = dat_sweden, aes(x = census, y = quota_1), color = "black") +
  geom_point(data = dat_norway, aes(x = census, y = quota_1), color = "gray70") +
  geom_line(data = dat_sweden, aes(x = swgrid, y = mod1_sweden$BUGSoutput$mean$b * swgrid), color = "bla
  geom_line(data = dat_norway, aes(x = nwgrid, y = mod1_norway$BUGSoutput$mean$b * nwgrid), color = "gra
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of family groups (year t - 1)",
       y = "Lynx quota (year t)")
```

```
## Warning in mod1_sweden$BUGSoutput$mean$b * swgrid: Recycling array of length 1 in array-vector arithm
##   Use c() or as.vector() instead.
```

```
## Warning in mod1_norway$BUGSoutput$mean$b * nwgrid: Recycling array of length 1 in array-vector arithm
##   Use c() or as.vector() instead.
```



Modèle 2

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

```
model2 <- function(){

  # Priors
  sigmaProc ~ dunif(0, 4)
  tauProc <- 1/sigmaProc^2
  b[1] ~ dnorm(0, 1/3000)
  b[2] ~ dnorm(0, 1/3000)
  # Process model
  for (t in 1:(nyears)) {
    mu[t] <- log(b[1] + b[2] * y[t])
#    mu[t] <- log(b[1] + b[2] * y[t]) * index[t]
#    index[t] <- - 1000 * step(y[t] + b[1] / b[2]) # step(x) = 1 if x >= 0
#    index[t] <- step(q[t]) # step(x) = 1 if x >= 0
#    mu[t] <- log(b[1] + b[2] * y[t])
    Hproc[t] <- max(0, mu[t])
    H[t] ~ dlnorm(Hproc[t], tauProc)

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

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

}
```

On prépare les données pour la Suède.

```
bugs.data <- list(
    nyears = 17,
    y = dat_sweden$census,
    q = dat_sweden$quota_1)
```

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

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

Allez zooh, on lance la machine!

```
mod2_sweden <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model2,
                  n.chains = bugs.chains,
                                                   n.thin = 10,
                                                   n.iter = 100000,
                                                   n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 17
##    Unobserved stochastic nodes: 20
##    Total graph size: 123
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

```
print(mod2_sweden, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a3
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
```

```
##            mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## b[1]       -68.612   9.619 -89.644 -67.971 -51.534 1.002  2900
## b[2]         1.023   0.116   0.813   1.018   1.268 1.002  2700
## sigmaProc    0.364   0.088   0.228   0.352   0.568 1.001 15000
## deviance   112.702   5.961 103.062 112.040 125.901 1.001  9100
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 17.8 and DIC = 130.5
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Les paramètres b sont estimés comme suit.

```
mod2_sweden$BUGSoutput$mean$b
```

```
## [1] -68.612431   1.023352
```

Le ratio se calcule comme suit.

```
- mod2_sweden$BUGSoutput$mean$b[1] / mod2_sweden$BUGSoutput$mean$b[2]
```

```
## [1] 67.04678
```

```
lm(q ~ y, data = bugs.data)
```

```
##
## Call:
## lm(formula = q ~ y, data = bugs.data)
##
## Coefficients:
## (Intercept)            y
##     -64.757        1.009
```

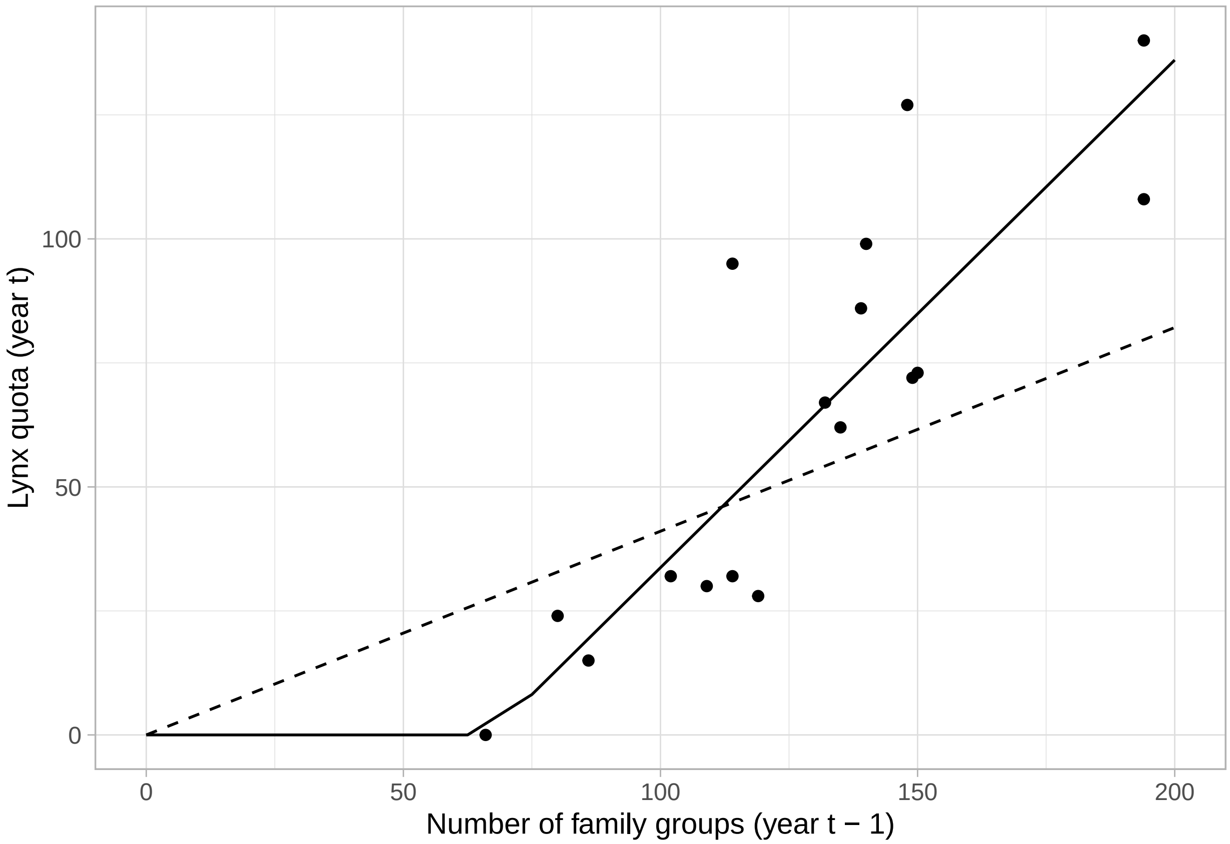
```
glm(q ~ y, data = bugs.data, family = "poisson")
```

```
##
## Call:  glm(formula = q ~ y, family = "poisson", data = bugs.data)
##
## Coefficients:
## (Intercept)            y
##     2.10350      0.01504
##
## Degrees of Freedom: 16 Total (i.e. Null);  15 Residual
## Null Deviance:       481.8
## Residual Deviance: 178.1     AIC: 276
```

Graphiquement, on obtient.

```
swgrid <- seq(0, 200, length.out = length(dat_sweden$census))
threshold <- - mod2_sweden$BUGSoutput$mean$b[1] / mod2_sweden$BUGSoutput$mean$b[2]
ggplot() +
  geom_point(data = dat_sweden, aes(x = census, y = quota_1), color = "black") +
  geom_line(data = dat_sweden, aes(x = swgrid, y = mod1_sweden$BUGSoutput$mean$b * swgrid), color = "bla
  geom_line(data = dat_sweden, aes(x = swgrid, y = if_else(swgrid < threshold, 0, (mod2_sweden$BUGSoutpu
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of family groups (year t - 1)",
       y = "Lynx quota (year t)")
```

```
## Warning in mod1_sweden$BUGSoutput$mean$b * swgrid: Recycling array of length 1 in array-vector arithm
##   Use c() or as.vector() instead.
```



Idem pour la Norvège. On prépare les données.

```
bugs.data <- list(
    nyears = 19,
    y = dat_norway$census,
    q = dat_norway$quota_1)
```

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

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

```
    list(
    )
}
```

Allez zooh, on lance la machine!

```
mod2_norway <- jags(data = bugs.data,
                  inits = bugs.inits,
                  parameters.to.save = bugs.monitor,
                  model.file = model2,
                  n.chains = bugs.chains,
                                                    n.thin = 10,
                                                    n.iter = 100000,
                                                    n.burnin = 50000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 22
##    Total graph size: 137
##
## Initializing model
```

```
threshold <- - mod2_norway$BUGSoutput$mean$b[1] / mod2_norway$BUGSoutput$mean$b[2]
threshold
```

```
## [1] 19.58581
```

Jetons un coup d'oeil aux estimations.

```
print(mod2_norway, intervals = c(2.5/100, 50/100, 97.5/100))
```

```
## Inference for Bugs model at "/var/folders/ln/jf2twlj12snbq000z6qq5y7m0000gn/T//RtmpIeNkmg/model16d2a5
##  3 chains, each with 1e+05 iterations (first 50000 discarded), n.thin = 10
##  n.sims = 15000 iterations saved
##           mu.vect sd.vect    2.5%     50%   97.5%  Rhat n.eff
## H[1]      131.318  10.932 110.887 130.946 153.720 1.001 15000
## H[2]      153.955  11.652 131.775 153.735 177.634 1.001 15000
## H[3]      134.550  10.921 113.821 134.153 156.964 1.001  4300
## H[4]      138.719  11.015 117.730 138.481 161.168 1.001 15000
## H[5]      121.296  10.414 102.067 120.993 142.676 1.001 15000
## H[6]      114.374  10.189  95.425 114.053 135.153 1.001  7500
## H[7]       85.018   8.301  69.611  84.723 101.979 1.001 15000
## H[8]       53.006   6.579  40.925  52.743  66.670 1.001 15000
## H[9]       51.790   6.475  39.977  51.486  65.168 1.001 11000
## H[10]      56.357   7.046  43.126  56.075  70.795 1.001 15000
## H[11]      79.454   8.165  64.074  79.203  96.009 1.001  7600
## H[12]     100.501   9.361  82.912 100.311 119.303 1.001 15000
## H[13]     121.726  10.213 102.436 121.501 142.276 1.001 15000
```

```
## H[14]     151.511  11.691 129.386 151.134 175.251 1.001 14000
## H[15]     171.037  12.639 147.399 170.473 196.635 1.001  7300
## H[16]     119.525  10.127 100.382 119.272 140.185 1.001 15000
## H[17]     119.857  10.147 100.986 119.574 140.519 1.001  5100
## H[18]      81.165   8.040  65.881  80.900  97.489 1.001  9900
## H[19]      86.970   8.493  71.470  86.603 104.472 1.001 15000
## b[1]      -45.821  26.110 -96.023 -46.851   8.081 1.001  5000
## b[2]        2.339   0.432   1.461   2.348   3.184 1.001  4300
## sigmaProc   0.233   0.053   0.150   0.227   0.355 1.001 13000
## deviance  142.407   6.336 132.013 141.787 156.508 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 = 20.1 and DIC = 162.5
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Les paramètres de régression sont estimés proches de la valeur qu'on trouve dans le Tableau 4.

```
mod2_norway$BUGSoutput$mean$b
```

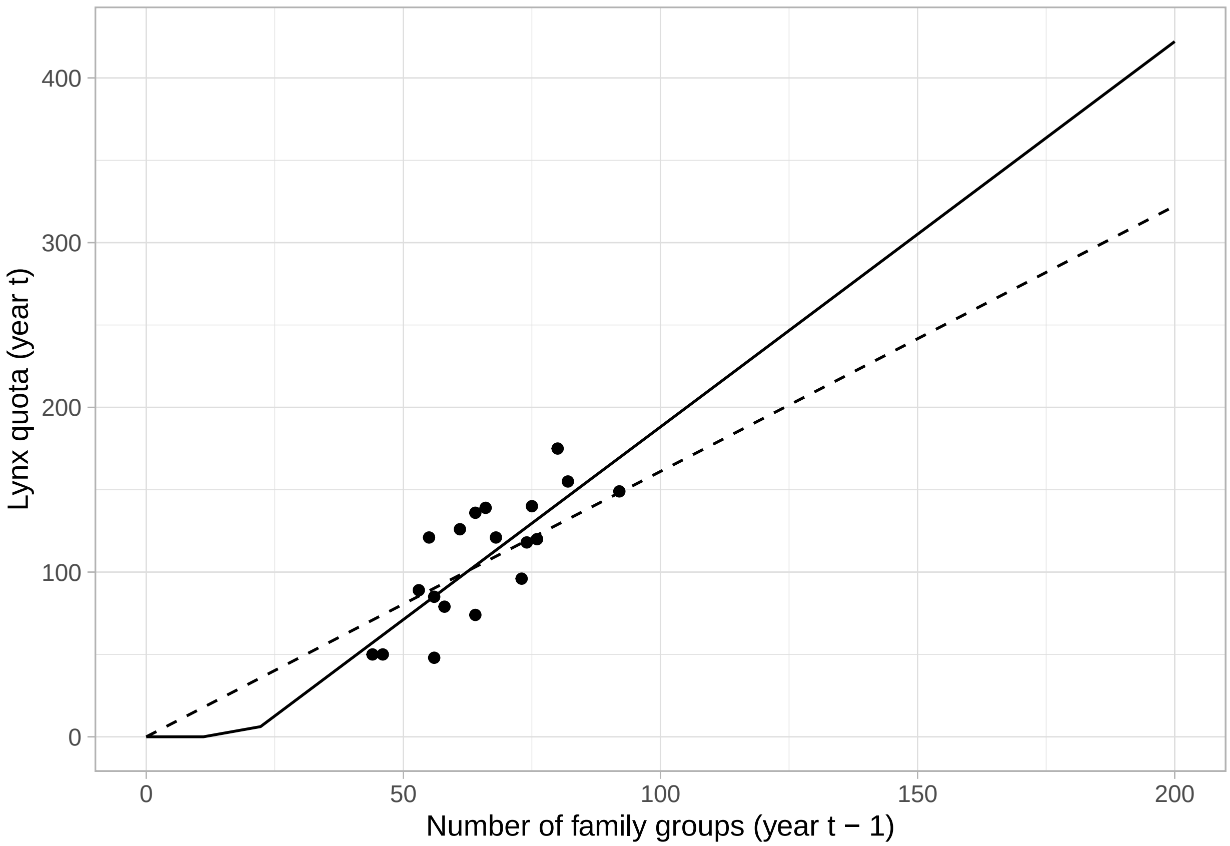
```
## [1] -45.820712   2.339485
```

Graphiquement, on obtient.

```
nwgrid <- seq(0, 200, length.out = length(dat_norway$census))

ggplot() +
  geom_point(data = dat_norway, aes(x = census, y = quota_1), color = "black") +
  geom_line(data = dat_norway, aes(x = nwgrid, y = mod1_norway$BUGSoutput$mean$b * nwgrid), color = "bla
  geom_line(data = dat_norway, aes(x = nwgrid, y = if_else(nwgrid < threshold, 0, (mod2_norway$BUGSoutpu
  expand_limits(x = 0, y = 0) +
  labs(x = "Number of family groups (year t - 1)",
       y = "Lynx quota (year t)")
```

```
## Warning in mod1_norway$BUGSoutput$mean$b * nwgrid: Recycling array of length 1 in array-vector arithm
##   Use c() or as.vector() instead.
```



400
300
200
100
0
Lynx quota (year t)
0
50
100
150
200
Number of family groups (year t − 1)