Calibration EMR - CMR et PVA loup

Olivier Gimenez

28/08/2020

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

```
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
```

```
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
```

```
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
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

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
```

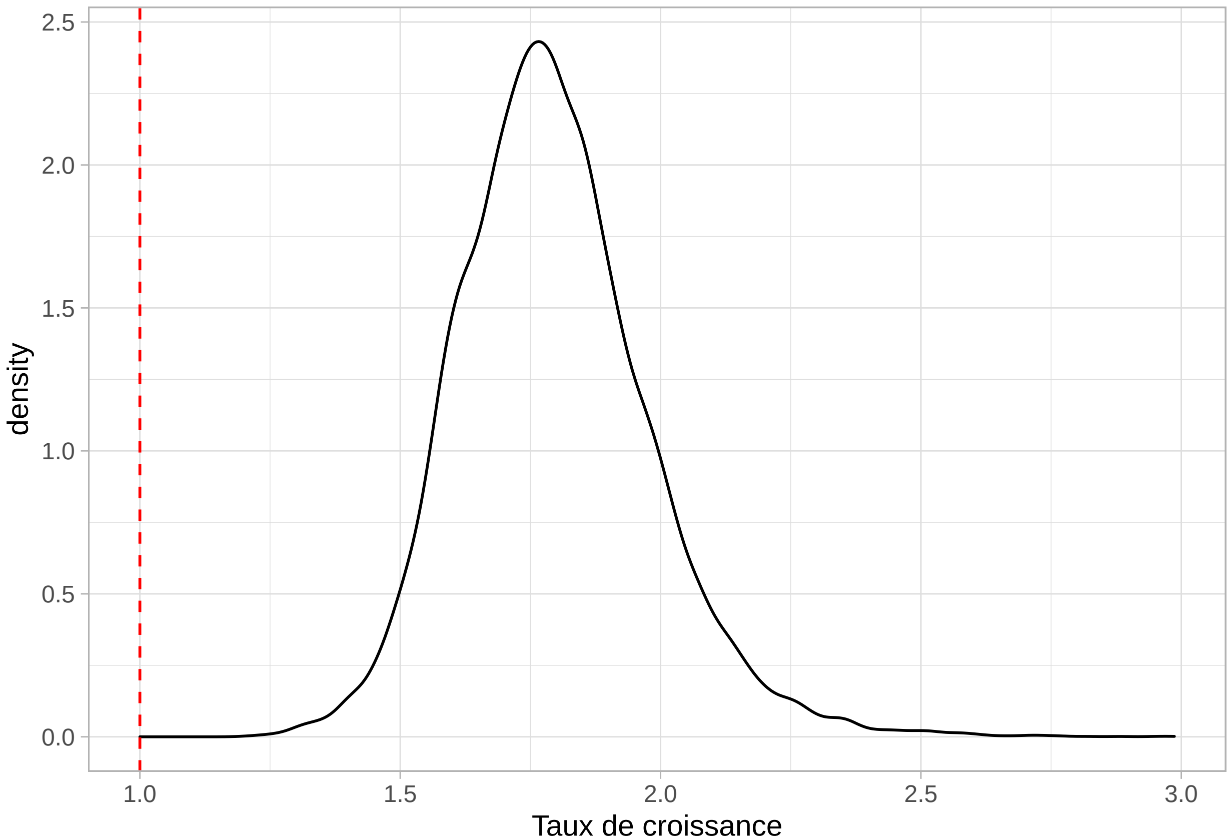
Jetons un coup d'oeil aux estimations.

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

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba1
##  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]      192.271  13.414 167.040 191.920 219.567 1.001 15000
## N[2]      190.865  13.282 165.508 190.511 217.776 1.001 15000
## N[3]      155.282  11.253 134.178 154.974 178.143 1.001 15000
## N[4]      146.586  11.284 125.655 146.203 169.886 1.001  8500
## N[5]      130.321  10.555 110.445 129.919 152.114 1.001 15000
## N[6]      111.142   9.915  92.568 110.801 131.532 1.001  6600
## N[7]       88.449   8.702  72.331  88.134 106.269 1.001 11000
## N[8]      115.140  10.323  95.866 114.772 136.367 1.001 15000
## N[9]      108.878   9.778  90.544 108.578 128.764 1.001 15000
## N[10]     103.584   9.424  86.196 103.325 122.888 1.001 15000
## N[11]     137.649  11.049 116.893 137.258 160.350 1.001 15000
## N[12]     150.902  10.993 130.452 150.571 173.215 1.001  4700
```

```
## N[13]     126.100   9.314 109.063 125.655 145.578 1.001 13000
## N[14]     144.183   9.788 126.119 143.828 164.323 1.001 14000
## N[15]     132.494   9.837 114.412 132.052 152.525 1.001 15000
## N[16]      97.744   6.134  86.743  97.446 110.526 1.001  5600
## N[17]      64.755   7.101  51.499  64.534  79.369 1.001  4400
## N[18]      92.172   8.890  75.482  91.880 110.408 1.001 15000
## N[19]     118.652   8.918 102.096 118.356 137.173 1.001 13000
## N[20]     106.611  10.240  87.608 106.297 127.726 1.001 15000
## lambda      1.793   0.182   1.477   1.781   2.192 1.001 15000
## sigmaProc   0.407   0.096   0.261   0.392   0.635 1.001  6200
## tauProc     7.036   3.169   2.481   6.503  14.721 1.001  6200
## deviance  154.850   6.456 144.156 154.241 169.152 1.001 10000
##
## 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 = 175.7
## 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")
```

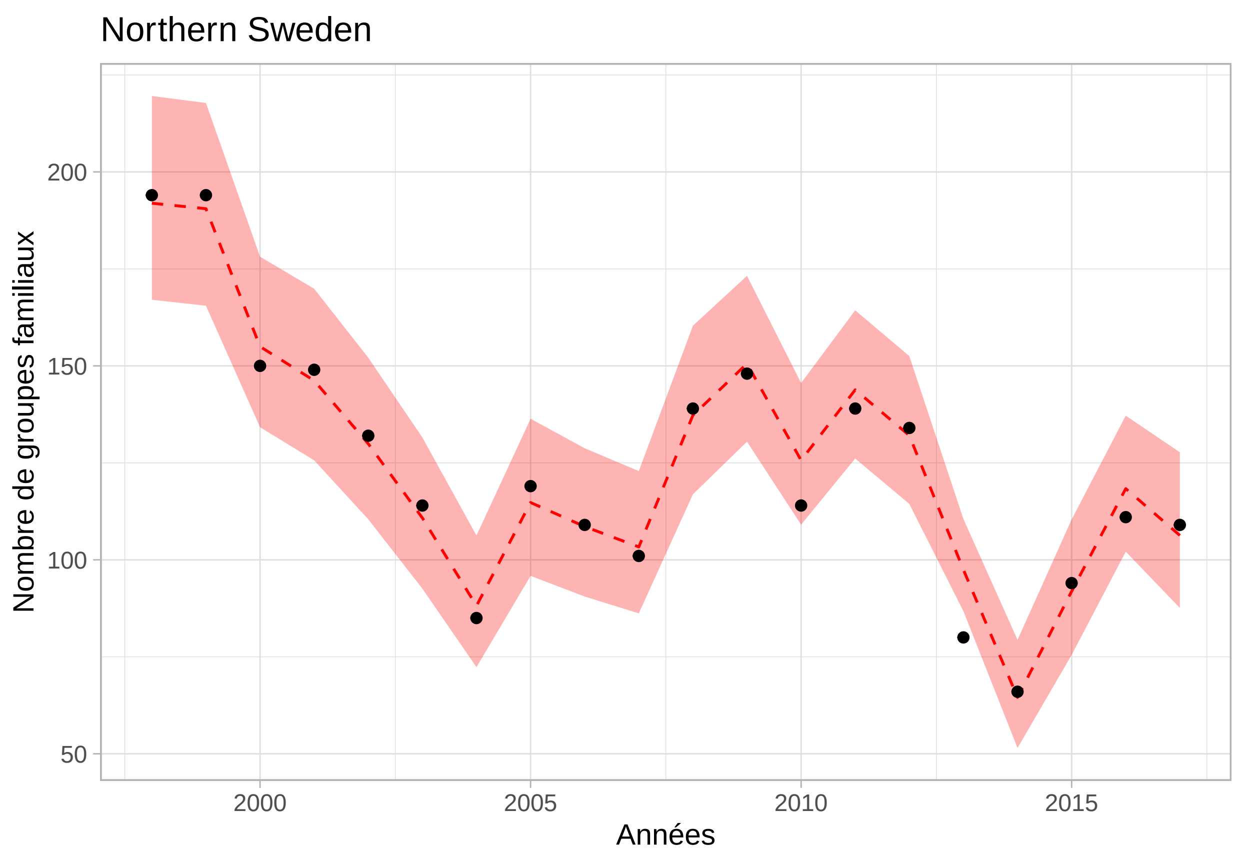


Ensuite les projections.

```
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")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
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
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
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!

```
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
```

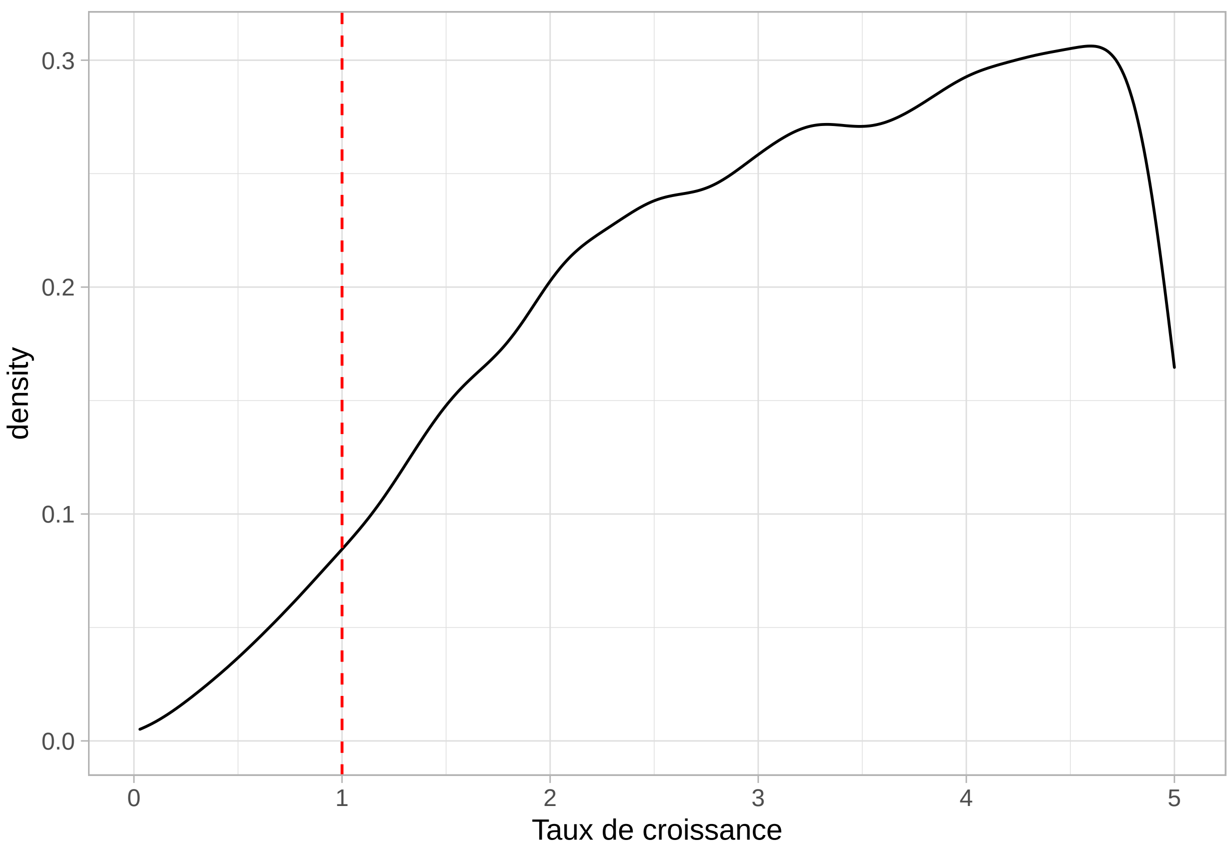
Jetons un coup d'oeil aux estimations.

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

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba6
##  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]       37.264   6.385 26.099 36.834 51.898 1.001  6500
## N[2]       46.855   7.105 34.343 46.369 62.795 1.001 10000
## N[3]       30.613   5.587 20.650 30.302 42.321 1.001 10000
## N[4]       35.179   5.565 25.424 34.757 47.311 1.001 10000
## N[5]       35.201   6.203 24.384 34.735 48.435 1.001 10000
## N[6]       19.745   4.494 11.799 19.451 29.423 1.001 10000
## N[7]       24.680   5.016 16.003 24.305 35.517 1.001 10000
## N[8]       26.087   4.819 17.325 25.676 36.728 1.001 10000
## N[9]       18.953   4.228 11.968 18.533 28.261 1.001 10000
## N[10]      21.842   4.328 13.876 21.569 31.263 1.001 10000
## N[11]      28.182   5.046 19.559 27.786 39.187 1.001  8500
## N[12]      36.322   5.488 25.565 36.003 47.720 1.001  3800
## N[13]      42.798   6.096 30.439 42.750 55.287 1.001 10000
## N[14]      50.767   7.484 36.764 50.834 65.441 1.001 10000
## N[15]      40.696   6.351 29.433 40.336 54.051 1.001 10000
## N[16]      36.548   6.124 25.575 36.152 49.751 1.001 10000
## N[17]      35.150   6.386 24.326 34.585 49.302 1.001 10000
## N[18]      28.266   5.736 18.261 27.706 40.430 1.001  8500
## N[19]      20.855   4.811 12.796 20.377 31.974 1.001 10000
## N[20]      30.910   5.288 20.892 30.744 41.994 1.001  8000
## N[21]      29.181   5.787 19.258 28.612 42.314 1.001 10000
## N[22]      30.512   5.544 20.777 30.149 42.309 1.001  9100
## lambda      3.199   1.171  0.834  3.308  4.927 1.001 10000
## sigmaProc   2.942   0.520  2.134  2.867  4.167 1.001  5100
```

```
## tauProc     0.126   0.042   0.058   0.122   0.220 1.001  5100
## deviance  138.283   6.758 127.124 137.653 153.386 1.001  3700
##
## 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 = 22.8 and DIC = 161.1
## 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")
```



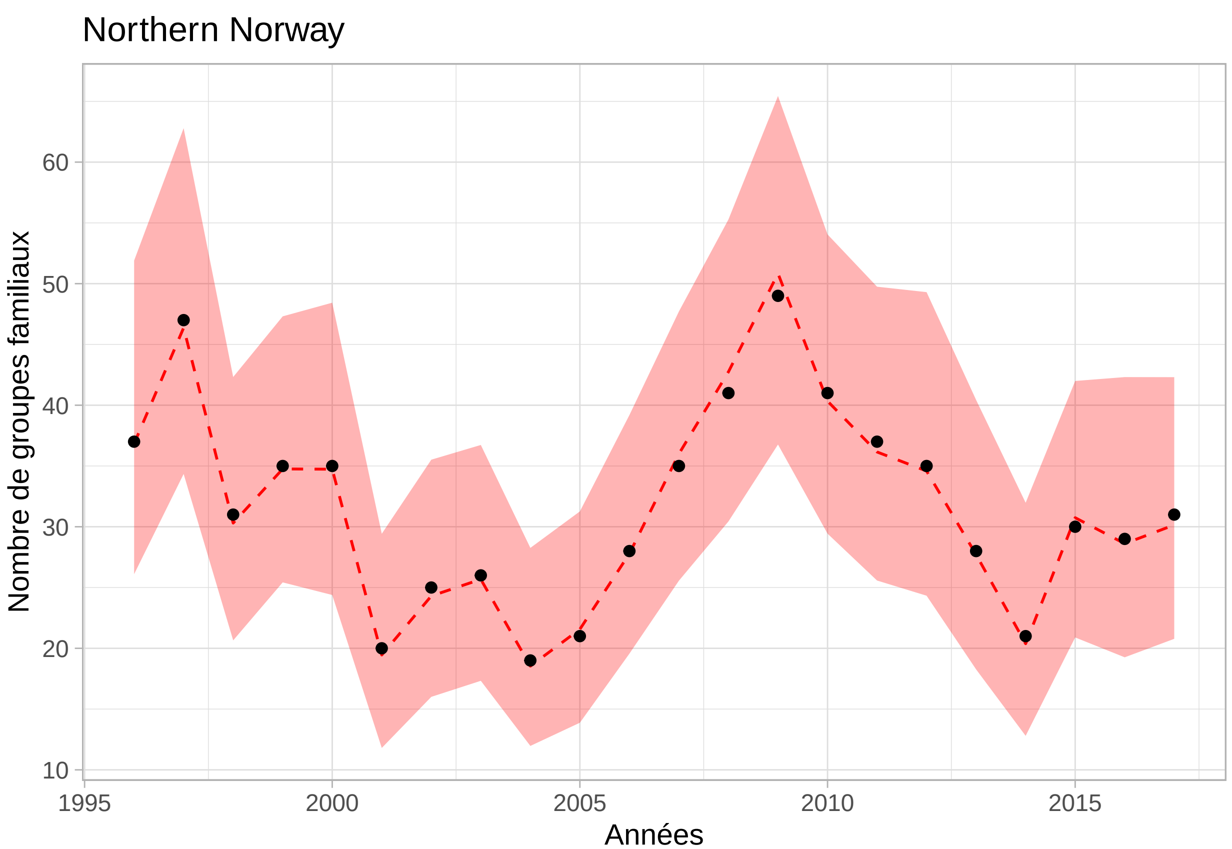
Ensuite les projections.

```
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")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
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") %>%
```

```r
  select(year, census, harvest) %>%
  group_by(year) %>%
  summarize(census = sum(census),
            harvest = sum(harvest)) -> dat1
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
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).

```r
bugs.monitor <- c("lambda", "sigmaProc","N", "tauProc")
bugs.chains <- 3
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
```

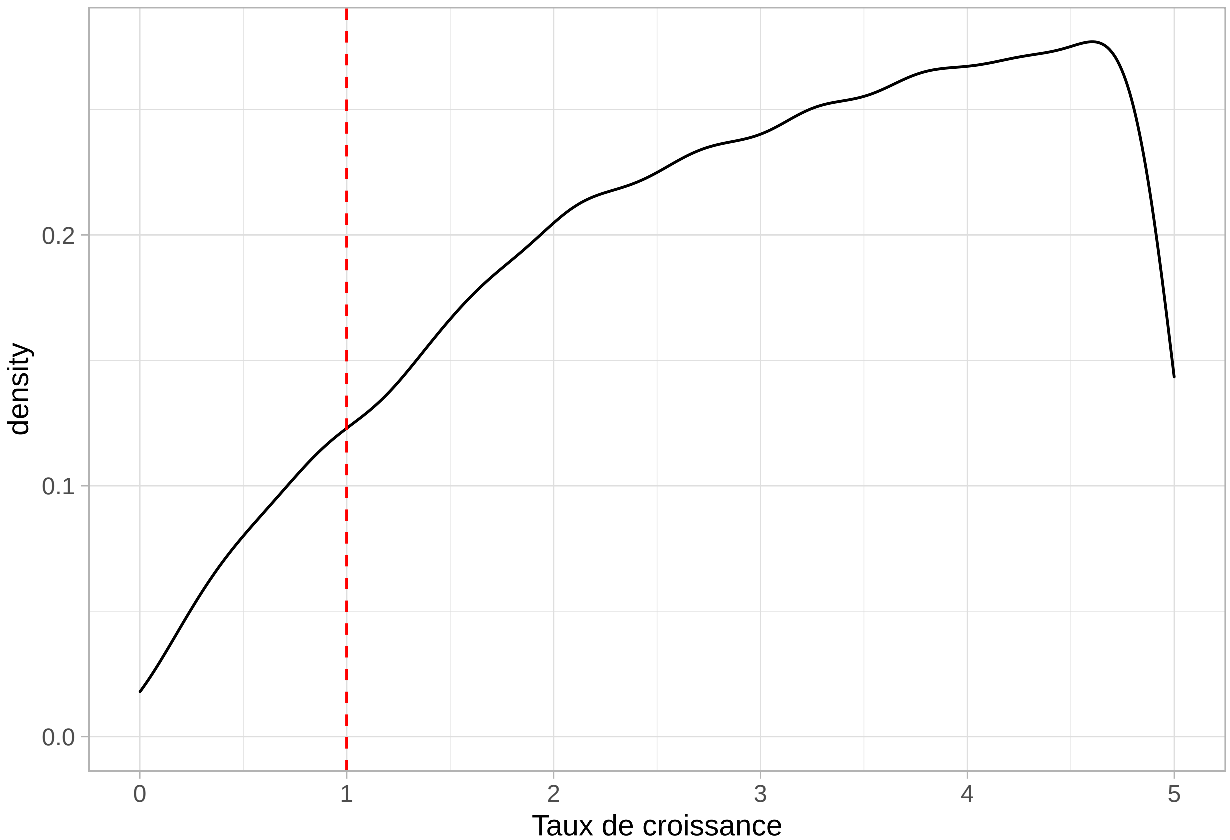
Jetons un coup d'oeil aux estimations.

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

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba4
##  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]       27.027   5.567  17.217  26.645  38.188 1.001 15000
```

```
## N[2]       34.026   5.807  23.022  34.106  45.649 1.001 15000
## N[3]       32.802   5.772  22.583  32.463  45.166 1.001  5900
## N[4]       39.715   6.334  28.303  39.305  53.038 1.001 11000
## N[5]       23.610   4.834  15.105  23.284  33.995 1.002  2500
## N[6]       31.804   5.789  21.530  31.376  44.694 1.001  8500
## N[7]       28.669   5.348  19.139  28.316  39.981 1.001  4900
## N[8]       18.651   4.292  11.224  18.336  27.895 1.001 15000
## N[9]       24.848   5.038  15.657  24.904  35.120 1.001 10000
## N[10]      33.657   5.432  23.398  33.341  45.158 1.001 15000
## N[11]      34.091   5.680  24.155  33.765  46.254 1.001 15000
## N[12]      36.983   5.673  26.359  36.654  49.070 1.001 14000
## N[13]      33.932   5.918  23.523  33.483  46.444 1.001  7700
## N[14]      40.706   6.490  28.990  40.279  54.691 1.001 15000
## N[15]      38.730   6.398  27.465  38.331  52.901 1.001 15000
## N[16]      34.616   5.893  24.075  34.276  47.042 1.001 15000
## N[17]      32.732   6.039  21.867  32.429  44.906 1.001 15000
## N[18]      28.988   5.656  19.222  28.497  41.683 1.001 15000
## N[19]      29.944   5.681  20.112  29.508  42.570 1.001 15000
## N[20]      28.678   5.344  19.065  28.349  40.104 1.001  7700
## N[21]      19.988   4.680  12.062  19.505  30.540 1.001 15000
## N[22]      25.667   5.051  16.708  25.304  36.413 1.001 15000
## lambda      3.013   1.273   0.489   3.140   4.910 1.001 12000
## sigmaProc   3.222   0.544   2.357   3.155   4.488 1.001 10000
## tauProc     0.104   0.033   0.050   0.100   0.180 1.001 10000
## deviance  138.223   6.754 126.969 137.569 152.993 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 = 22.8 and DIC = 161.0
## 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")
```

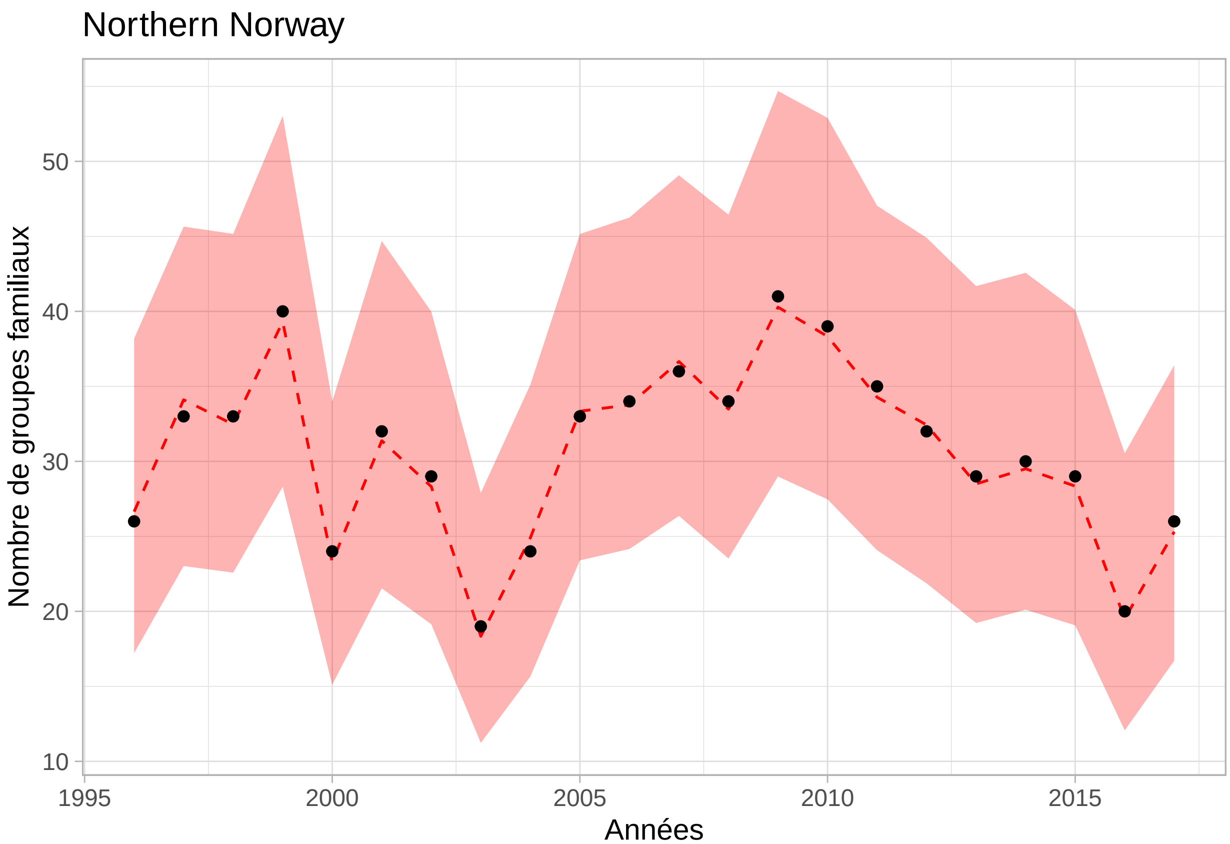


Ensuite les projections.

```
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")
```

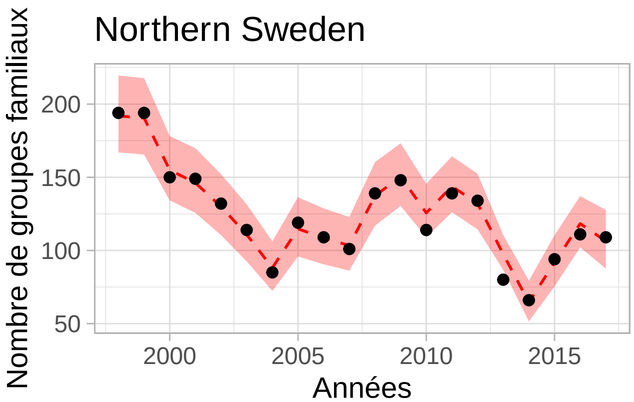
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

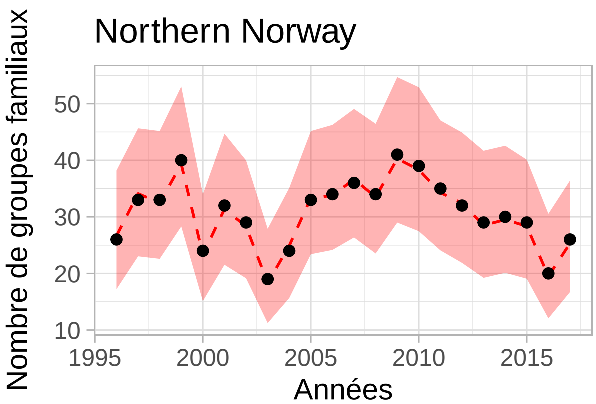
```
southern_norway
```

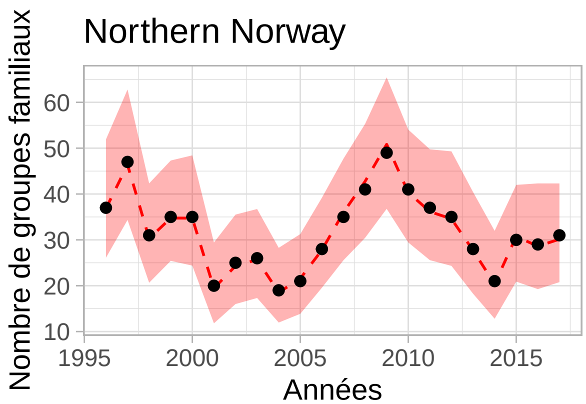


Tout ensemble - Figure 3 ou presque

```
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
```

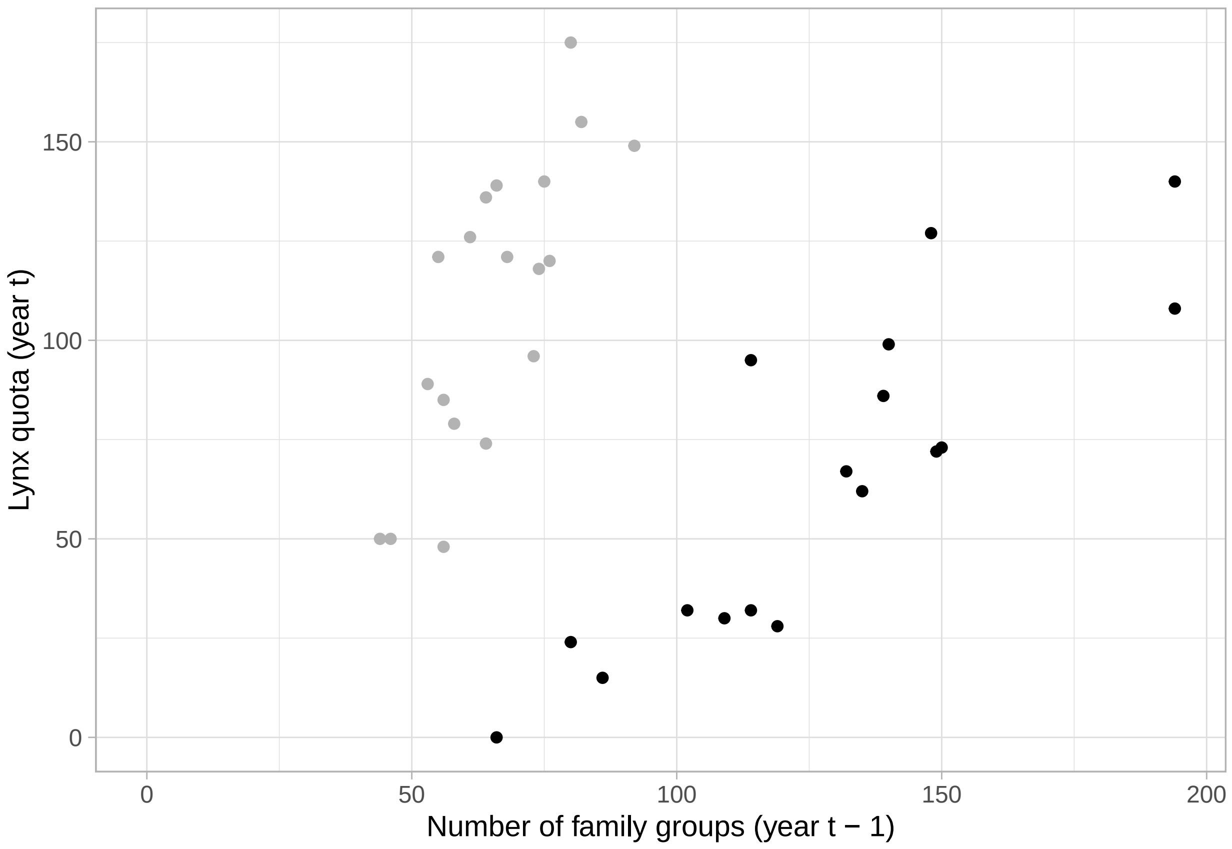
```r
dat %>%
  filter(country == "2") %>%
  select(year, census, quota_1) -> dat_norway

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
```

```r
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.

```r
model1 <- function(){

  # Priors
```

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

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

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

}
```

On prépare les données 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: 106
##
## 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/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba2
##  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]      138.820  11.644 116.906 138.461 162.411 1.001 15000
## H[2]      107.162  10.212  87.998 106.909 128.201 1.001  6000
## H[3]       72.540   8.450  56.840  72.234  89.975 1.001 15000
## H[4]       71.636   8.387  56.162  71.313  88.911 1.001  6800
## H[5]       66.645   8.147  51.502  66.326  83.318 1.001 15000
## H[6]       32.686   5.598  22.586  32.392  44.500 1.001  4500
## H[7]       16.515   3.928   9.734  16.237  25.075 1.001 15000
## H[8]       28.987   5.205  19.569  28.689  39.958 1.001 15000
## H[9]       30.665   5.377  20.992  30.360  41.953 1.001 12000
## H[10]      32.430   5.552  22.484  32.116  44.248 1.001 12000
## H[11]      97.887   9.821  79.558  97.614 118.025 1.001 15000
## H[12]     125.485  11.281 104.591 125.145 148.454 1.001 15000
## H[13]      93.478   9.556  75.486  93.209 113.067 1.001  7000
## H[14]      85.160   9.131  68.351  84.668 104.115 1.001 15000
## H[15]      61.757   7.742  47.618  61.433  77.647 1.001 15000
## H[16]      24.568   4.820  16.143  24.252  34.824 1.001  4100
## H[17]       3.816   2.041   0.856   3.497   8.672 1.001  9600
## b           0.402   0.079   0.264   0.395   0.578 1.001 15000
## sigmaProc   0.775   0.200   0.479   0.744   1.261 1.001 11000
## deviance  117.394   6.889 105.810 116.757 132.765 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 = 23.7 and DIC = 141.1
## 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] 0.4016313
```

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!

```
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: 118
##
## Initializing model
```

Jetons un coup d'oeil aux estimations.

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

```
## Inference for Bugs model at "/var/folders/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba2
##  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]      131.554  10.999 110.967 131.229 154.279 1.001  5700
## H[2]      152.424  11.889 130.139 152.145 176.567 1.001 15000
## H[3]      134.849  11.144 114.036 134.445 157.925 1.001 14000
## H[4]      137.562  11.180 116.786 137.159 160.529 1.001 15000
## H[5]      121.949  10.518 102.434 121.539 143.839 1.001 15000
## H[6]      116.075  10.206  97.155 115.851 137.242 1.001 15000
## H[7]       85.564   8.646  69.626  85.283 103.406 1.001 15000
## H[8]       54.541   6.714  41.857  54.317  68.301 1.001 15000
## H[9]       54.072   6.667  41.707  53.782  67.759 1.001 15000
## H[10]      55.497   7.089  42.489  55.279  69.916 1.001 15000
## H[11]      78.110   8.295  62.628  77.861  95.160 1.001 15000
## H[12]      98.468   9.268  81.272  98.154 117.447 1.001 13000
## H[13]     119.852  10.236 100.565 119.638 140.342 1.001 15000
## H[14]     148.542  11.684 126.391 148.401 171.981 1.001 11000
## H[15]     169.836  12.555 146.086 169.553 195.171 1.001 15000
## H[16]     117.773  10.308  98.610 117.329 138.835 1.001 15000
```

```
## H[17]     119.205  10.283 100.165 118.890 140.343 1.001  6200
## H[18]      81.017   8.326  65.595  80.735  98.042 1.001  6900
## H[19]      88.024   8.613  71.970  87.767 105.559 1.001 11000
## b           1.560   0.101   1.360   1.560   1.758 1.001 15000
## sigmaProc   0.262   0.058   0.170   0.255   0.397 1.001 15000
## deviance  142.562   6.433 132.069 141.859 156.892 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 = 20.7 and DIC = 163.3
## 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
```

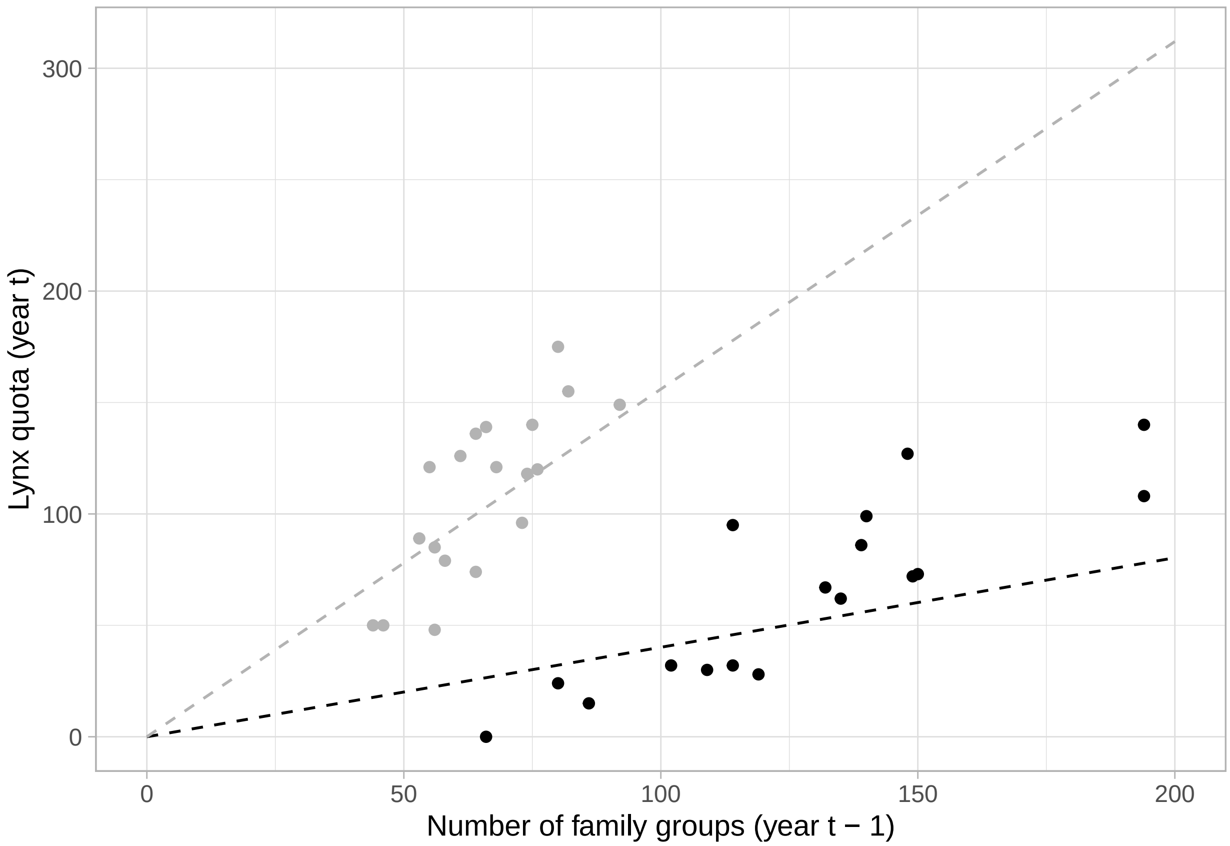
```
## [1] 1.559921
```

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, 1000)
  b[2] ~ dnorm(0, 1000)
  # 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: 122
##
## 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/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba3
##  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]         0.001   0.032  -0.062   0.001   0.063 1.001  9200
## b[2]         0.087   0.024   0.042   0.086   0.136 1.001  9900
## sigmaProc    1.956   0.485   1.210   1.885   3.114 1.001 15000
## deviance   111.594   5.872 101.977 110.971 124.688 1.001  9800
##
## 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 = 17.2 and DIC = 128.8
## 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] 0.0005412535 0.0865190161
```

Le ratio se calcule comme suit.

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

```
## [1] -0.006255891
```

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

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

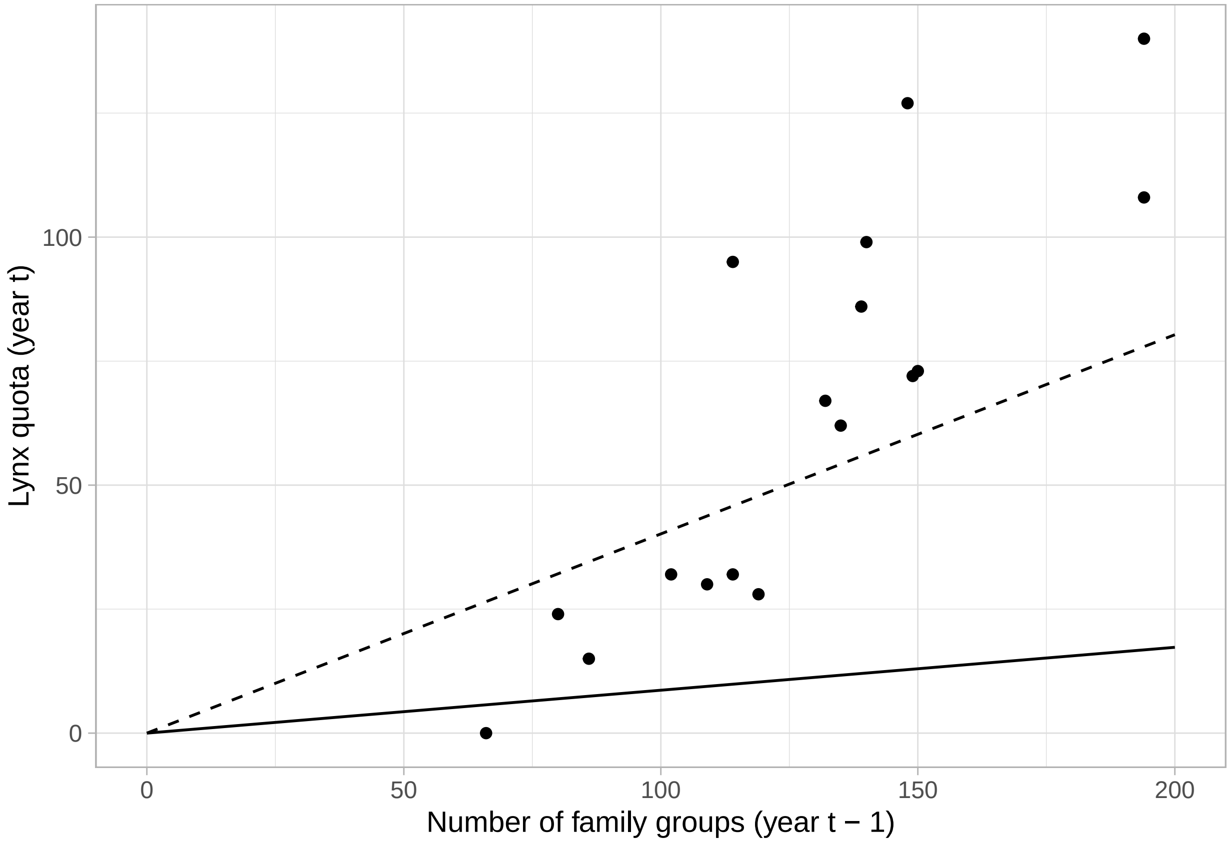
```
glm(q ~ y, data = bugs.data, family = "poisson")
```

```
##
## 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$census))
ggplot() +
  geom_point(data = dat_sweden, aes(x = census, y = quota_1), color = "black") +
  geom_line(data = dat_sweden, aes(x = swgrid, y = mod1_sweden$BUGSoutput$mean$b * swgrid), color = "bla
  geom_line(data = dat_sweden, aes(x = swgrid, y = (mod2_sweden$BUGSoutput$mean$b[1] +  mod2_sweden$BUGS
  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: 136
##
## Initializing model
```

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/r7/j0wqj1k95vz8w44sdxzm986c0000gn/T//RtmpZaM5Ca/model3dba5
##  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]     135.692  11.600 113.919 135.380 159.185 1.001 15000
## H[2]     154.654  12.376 131.473 154.214 180.212 1.001 15000
## H[3]     138.665  11.766 116.793 138.306 162.531 1.001 15000
## H[4]     139.543  11.704 117.552 139.355 163.182 1.001 15000
## H[5]     125.724  11.270 104.445 125.397 148.736 1.001 15000
## H[6]     120.823  10.938 100.586 120.514 143.319 1.001 10000
## H[7]      84.684   9.203  67.455  84.328 103.645 1.001 15000
## H[8]      49.716   7.025  36.852  49.398  64.364 1.001 15000
## H[9]      49.721   7.075  36.817  49.358  64.835 1.001 12000
## H[10]     47.800   6.980  35.223  47.436  62.252 1.001 15000
## H[11]     73.576   8.579  57.782  73.234  91.432 1.001 13000
## H[12]     95.763   9.684  77.783  95.426 115.330 1.001 14000
## H[13]    119.511  10.990  99.042 119.257 141.878 1.001 15000
## H[14]    148.676  12.280 125.624 148.415 173.373 1.001 15000
## H[15]    174.670  13.356 149.538 174.325 201.547 1.001 15000
## H[16]    117.763  10.838  97.659 117.430 140.327 1.001 11000
## H[17]    120.656  11.072  99.777 120.354 143.377 1.001 15000
## H[18]     78.717   8.875  62.031  78.453  97.045 1.001  7800
## H[19]     88.776   9.418  71.326  88.446 108.044 1.001 12000
## b[1]       0.002   0.032  -0.061   0.002   0.064 1.001 15000
```

```
## b[2]        0.082   0.023   0.040   0.081   0.131 1.001  9600
## sigmaProc   3.070   0.467   2.202   3.065   3.910 1.001 15000
## deviance  141.936   6.244 131.657 141.298 156.111 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 = 19.5 and DIC = 161.4
## 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.

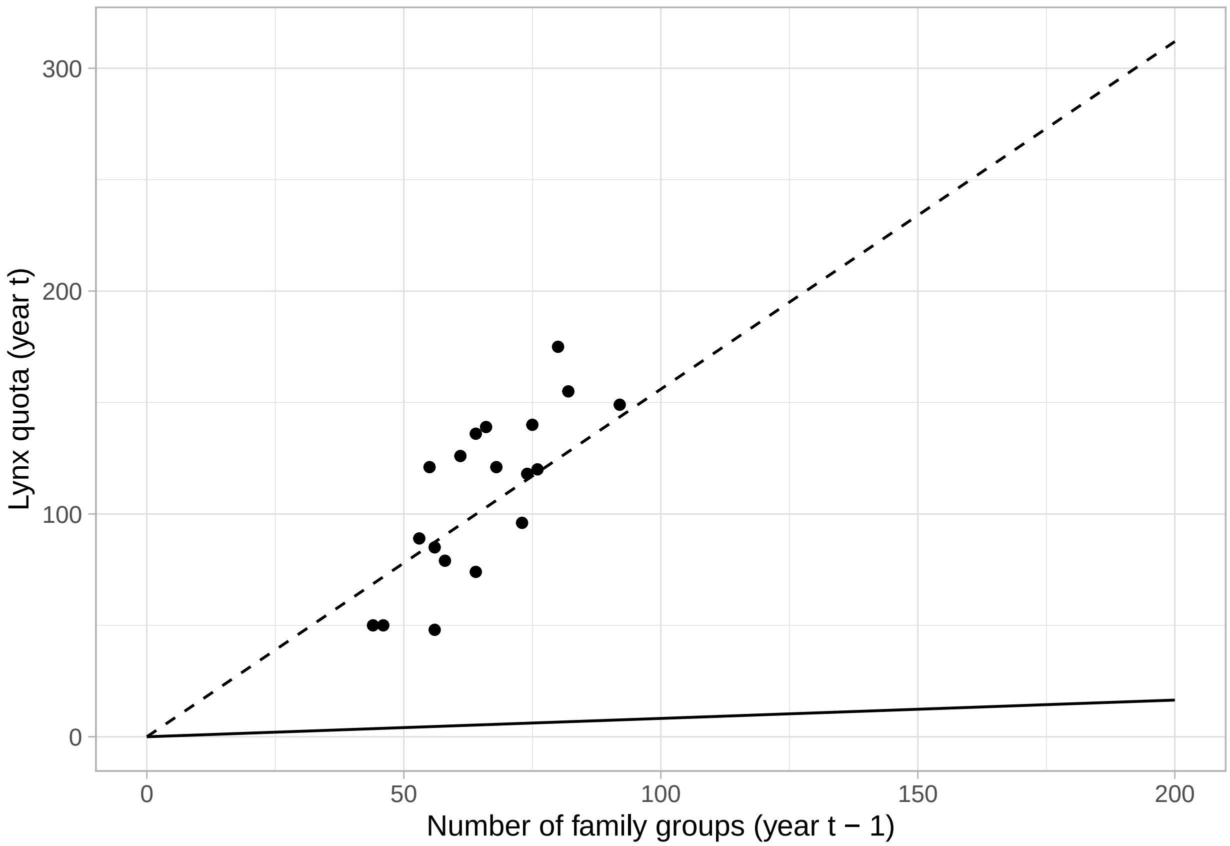
```
mod2_norway$BUGSoutput$mean$b
```

```
## [1] 0.001526982 0.082346563
```

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 = (mod2_norway$BUGSoutput$mean$b[1] +  mod2_norway$BUGS
  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.
```



Je n'arrive pas reproduire ce satané modèle à seuil avec l'intercept non-nul, les lignes en trait plein de la figure 4 du papier. Grrrr. On arrive aux estimations du Tableau 3 via une simple régression hein, donc c'est pas bien grave.