Statistique bayésienne avec R

Exercice Capture-Mark-Recapture

Julien JACQUES

Nous allons refaire l'analyse présentée dans la section 9.2 du livre suivant

http://sirs.agrocampus-ouest.fr/bayes_V2/index.html

Les données sont

- y1 : nombre de poissons piégés
- y2+y3 : poissons capturés non relachés
- y4=y1-(y2+y3) : nombre de poissons marqués et relachés
- y5 : nombre de poissons marqués recapturés
- y6 : nombre de poissons non marqués recapturés

Years	y_1	y_2	y_3	y_4	y_5	y_6
1984	167	10	3	154	12	10
1985	264	37	11	216	21	4
1986	130	28	9	93	5	4
1987	16	3	1	12	2	22
1988	226	35	8	183	12	0
1989	235	31	5	199	56	0
1990	15	4	4	7	2	15
1991	44	0	0	44	23	1
1992	31	10	1	20	4	5
1993	100	17	2	81	4	3
1994	32	12	2	18	1	4
1995	109	6	1	102	39	7
1996	70	13	2	55	25	57
1997	56	19	3	34	12	3
1998	34	3	1	30	6	30
1999	154	5	1	148	13	22
2000	53	0	0	53	4	33

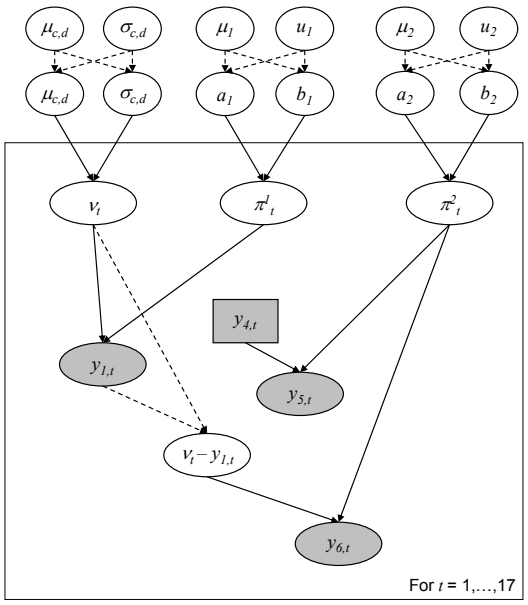
On suppose que

$$y_1 \sim \mathcal{B}(n, p)$$

où :

- n est la taille de la population, que l'on cherche à connaître
- p est l'efficacité de pièggeage

Le modèle proposé est le suivant



Analyse bayesienne

Commençons par charger les données

On commence par définir les données

```
data <- list(y1=CMRsurvey$y1,y2_plus_y3=CMRsurvey$y2+CMRsurvey$y3,y4=CMRsurvey$y4,y5=CMRsurvey$y5,
             y6=CMRsurvey$y6,N = nrow(CMRsurvey))
```

puis on définit les initialisations et les modèles

```
inits1 <- list(list(mu1=.5,logu1=5,mu2=.5,logu2=5,mu3=1000,logs3=2),
               list(mu1=.7,logu1=3,mu2=.4,logu2=6,mu3=2000,logs3=1.9))
library(rjags)
m1 <- jags.model('CMR.txt', data = data, inits = inits1, n.chains = 2, quiet=TRUE)
```

Puis on lance les itérations MCMC

```
update(m1,10000,progress.bar="none")
mcmc1 <- coda.samples(m1, variable.names = c("pi1", "nu"), n.iter = 10000,progress.bar="none")
```

On vérifie que les chaines ont bien convergées.

```
gelman.diag(mcmc1)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## nu[1]         1.00       1.00
## nu[2]         1.00       1.00
## nu[3]         1.01       1.02
## nu[4]         1.01       1.02
## nu[5]         1.03       1.04
## nu[6]         1.00       1.00
## nu[7]         1.01       1.03
## nu[8]         1.00       1.00
```

```
## nu[9]          1.00       1.01
## nu[10]         1.03       1.09
## nu[11]         1.06       1.17
## nu[12]         1.00       1.00
## nu[13]         1.00       1.01
## nu[14]         1.00       1.01
## nu[15]         1.03       1.06
## nu[16]         1.01       1.06
## nu[17]         1.00       1.00
## pi1[1]         1.00       1.00
## pi1[2]         1.00       1.00
## pi1[3]         1.01       1.03
## pi1[4]         1.00       1.00
## pi1[5]         1.02       1.02
## pi1[6]         1.00       1.00
## pi1[7]         1.00       1.01
## pi1[8]         1.00       1.00
## pi1[9]         1.00       1.00
## pi1[10]        1.01       1.05
## pi1[11]        1.02       1.11
## pi1[12]        1.00       1.00
## pi1[13]        1.00       1.00
## pi1[14]        1.00       1.01
## pi1[15]        1.00       1.02
## pi1[16]        1.01       1.05
## pi1[17]        1.00       1.00
##
## Multivariate psrf
##
## 1.06
```

```
#autocorr.plot(mcmc1[[1]])
```

Regardons les estimations

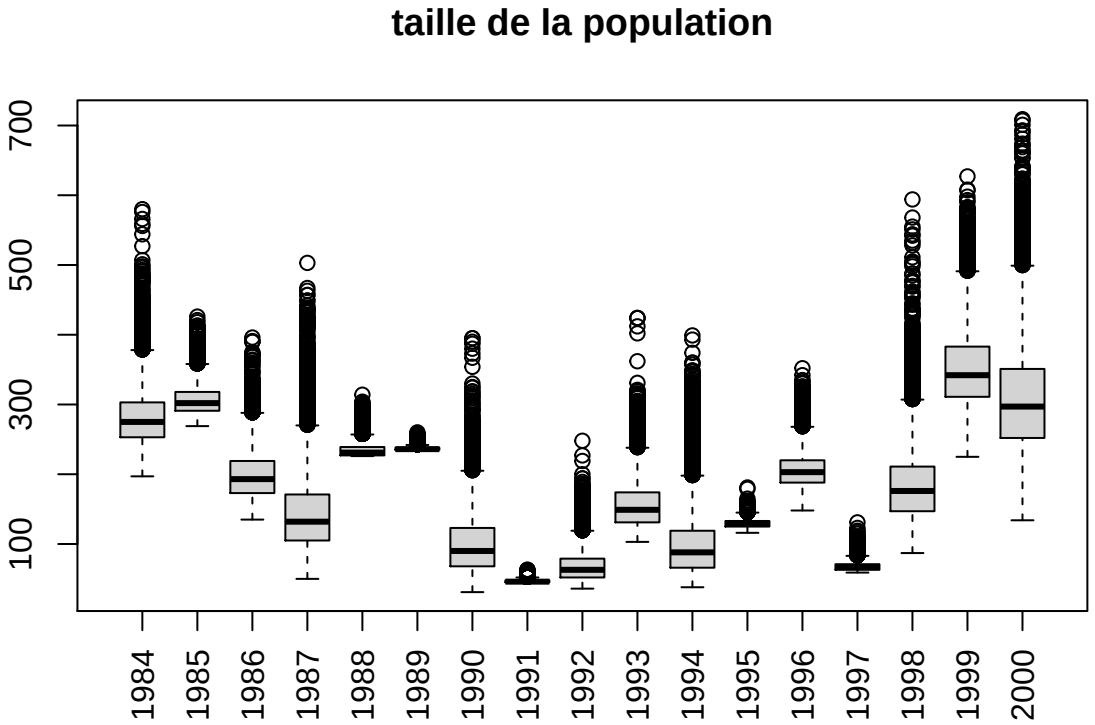
```
summary(mcmc1)
```

```
##
## Iterations = 11001:21000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean       SD  Naive SE Time-series SE
## nu[1]   281.5555 42.59505 3.012e-01      1.7616141
## nu[2]   306.4671 21.17493 1.497e-01      0.7053783
## nu[3]   197.2722 36.12763 2.555e-01      1.5825260
## nu[4]   142.7585 57.00217 4.031e-01      3.3419959
## nu[5]   235.3233 12.24693 8.660e-02      0.4770473
## nu[6]   236.9981  2.86824 2.028e-02      0.0567668
## nu[7]    98.5517 43.16853 3.052e-01      2.2452298
## nu[8]    47.1339  2.35003 1.662e-02      0.0369538
```

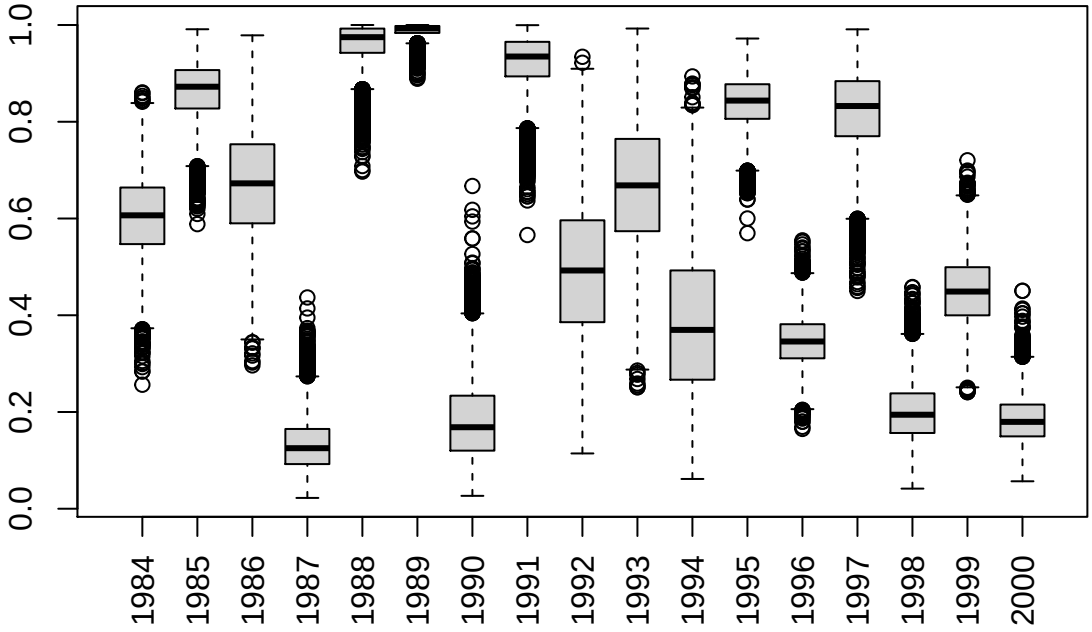
```
## nu[9]    69.8650 25.18757 1.781e-01      1.1341165
## nu[10]  159.3364 37.88944 2.679e-01      1.8810645
## nu[11]   94.8060 44.95386 3.179e-01      2.6594901
## nu[12]  129.6122  6.73887 4.765e-02      0.1144670
## nu[13]  205.2638 24.63974 1.742e-01      0.6664333
## nu[14]   68.0461  6.75032 4.773e-02      0.1454915
## nu[15]  182.4066 51.11214 3.614e-01      2.4528618
## nu[16]  356.1538 59.61130 4.215e-01      2.6242054
## nu[17]  308.1931 78.75334 5.569e-01      4.1393316
## pi1[1]    0.6050  0.08775 6.205e-04      0.0034012
## pi1[2]    0.8637  0.05915 4.183e-04      0.0019393
## pi1[3]    0.6781  0.11497 8.130e-04      0.0048639
## pi1[4]    0.1343  0.05512 3.897e-04      0.0020914
## pi1[5]    0.9597  0.04617 3.265e-04      0.0017839
## pi1[6]    0.9886  0.01371 9.695e-05      0.0002644
## pi1[7]    0.1863  0.08181 5.785e-04      0.0032272
## pi1[8]    0.9228  0.05560 3.932e-04      0.0008092
## pi1[9]    0.4894  0.14614 1.033e-03      0.0054959
## pi1[10]   0.6567  0.13497 9.544e-04      0.0061553
## pi1[11]   0.4043  0.15802 1.117e-03      0.0071064
## pi1[12]   0.8398  0.05212 3.686e-04      0.0008450
## pi1[13]   0.3478  0.05149 3.641e-04      0.0011042
## pi1[14]   0.8237  0.08334 5.893e-04      0.0016652
## pi1[15]   0.2030  0.05899 4.171e-04      0.0022201
## pi1[16]   0.4440  0.07294 5.158e-04      0.0028414
## pi1[17]   0.1848  0.04894 3.461e-04      0.0020030
##
## 2. Quantiles for each variable:
##
##              2.5%       25%      50%      75%    97.5%
## nu[1]   220.00000 251.00000 274.0000 304.0000 385.0250
## nu[2]   277.00000 291.00000 302.0000 317.0000 359.0000
## nu[3]   148.00000 171.00000 190.0000 217.0000 284.0000
## nu[4]    71.00000 104.00000 130.0000 168.0000 292.0000
## nu[5]   226.00000 227.00000 231.0000 239.0000 268.0000
## nu[6]   235.00000 235.00000 236.0000 238.0000 245.0000
## nu[7]    46.00000  67.00000  88.0000 118.0000 212.0000
## nu[8]    45.00000  45.00000  46.0000  48.0000  53.0000
## nu[9]    42.00000  53.00000  63.0000  80.0000 136.0000
## nu[10]  112.00000 132.00000 151.0000 177.0000 257.0000
## nu[11]   45.00000  63.00000  83.0000 113.0000 216.0250
## nu[12]  119.00000 125.00000 129.0000 133.0000 145.0000
## nu[13]  166.00000 188.00000 202.0000 219.0000 262.0000
## nu[14]   60.00000  63.00000  66.0000  71.0000  85.0000
## nu[15]  113.00000 146.00000 174.0000 208.0000 305.0000
## nu[16]  268.00000 314.00000 346.5000 388.0000 503.0000
## nu[17]  193.00000 253.00000 295.0000 349.0000 509.0000
## pi1[1]    0.42701   0.54633   0.6074   0.6657   0.7712
## pi1[2]    0.72779   0.82815   0.8719   0.9073   0.9567
## pi1[3]    0.45165   0.59446   0.6824   0.7637   0.8844
## pi1[4]    0.04999   0.09373   0.1267   0.1660   0.2638
## pi1[5]    0.83407   0.94308   0.9758   0.9927   0.9999
## pi1[6]    0.95082   0.98415   0.9933   0.9981   1.0000
## pi1[7]    0.06463   0.12460   0.1739   0.2354   0.3751
```

```
## pi1[8]    0.78469   0.89379   0.9354   0.9648   0.9928
## pi1[9]    0.21671   0.38301   0.4892   0.5945   0.7725
## pi1[10]   0.38407   0.56289   0.6619   0.7562   0.8994
## pi1[11]   0.14443   0.28148   0.3902   0.5160   0.7305
## pi1[12]   0.72669   0.80663   0.8442   0.8776   0.9282
## pi1[13]   0.25081   0.31229   0.3469   0.3816   0.4517
## pi1[14]   0.63277   0.77305   0.8351   0.8863   0.9514
## pi1[15]   0.10404   0.16009   0.1970   0.2403   0.3324
## pi1[16]   0.30053   0.39431   0.4438   0.4938   0.5869
## pi1[17]   0.10042   0.15002   0.1806   0.2155   0.2914
```

```
X=as.matrix(mcmc1[[1]])
boxplot(X[,1:17],main='taille de la population',xaxt = "n")
axis(side =1, at=1:17, labels = 1984:2000, las=3)
```




```
boxplot(X[,18:34],main='efficacité du piégeage',xaxt = "n")
axis(side =1, at=1:17, labels = 1984:2000, las=3)
```


efficacité du piégeage
0.0
0.2
0.4
0.6
0.8
1.0
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000