Modelos de marcação e recaptura: populações fechadas

Contents

- Arquivo em pdf
- Arquivo em markdown (para executar os comandos no R studio)

Preparação

Vamos usar o pacote *RMark*, que é um pacote do R para usar o programa MARK. Siga as instruções deste sítio para instalar o MARK para uso pelo pacote: (http://www.phidot.org/software/mark/rmark/).

Abra o R e carregue o pacote

```
library(RMark)
```

Usaremos dados de registro fotográfico de indivíduos do boto cinza (*Sotalia guianensis*) em 11 ocasiões. Aqui há mais informações sobre este caso de estudo.

Os dados estão no formato nativo do MARK (*.inp*). Use os comandos abaixo para importá-lo para o R:

```
## Link dos dados na página da disciplina
url <- "http://ecologia.ib.usp.br/bie5703/lib/exe/fetch.php?media=roteiros:botos_2002.inp"
## Importa arquivo inp
boto2002 <- convert.inp(url)
```

Ajuste dos modelos

Processamento dos dados

O primeiro passo é usar a função `process.data` para criar um objeto com as informações que o Mark usa para ajustar o modelo. Uma delas é o tipo de modelo, que é indicado no argumento `model`.

Para o modelo de populações fechada sem heterogeneidade e de verossimilhança não condicionada este argumento é `model="Closed"`:

```
boto <- process.data(data=boto2002, model="Closed")
```

E para o modelo com heterogeneidade o argumento é `model="FullHet"`

```
botoH <- process.data(data=boto2002, model="FullHet")
```

Ajuste dos modelos sem heterogeneidade

Para ajustar os modelos, crie listas que especificam a fórmula de cada termo. No modelo `Closed` os nomes parâmetros que podem variar são `p` (probabilidade da primeira captura), `c` (probabilidade de recaptura). O objeto criado na seção acima tem uma covariável de tempo chamada `time`, que então pode ser usado nas fórmulas:

```
## Fórmulas estatísticas para cada parâmetro do modelo sem heterogeneidade
## p e c constantes mas diferentes
t.dot <- list(formula=~1)
## p=c contantes (use o argumento share=TRUE)
t.dotshared=list(formula=~1,share=TRUE)
## Parametros dependem do tempo
t.time <- list(formula=~time)
## Parametro p=c dependem do tempo
t.timeshared <- list(formula=~time, share=TRUE)
```

E usamos a função `mark` para fazer os ajuste:

```
boto.M0 <- mark(boto, model.parameters=list(p=t.dotshared))
boto.Mb <- mark(boto, model.parameters=list(c=t.dot, p=t.dot))
```

```
##
## Note: only 2 parameters counted of 3 specified parameters
##
## AICc and parameter count have been adjusted upward
```

```
boto.Mt <- mark(boto, model.parameters=list(p=t.timeshared))
boto.Mtb <- mark(boto, model.parameters=list(c=t.time, p=t.time))
```

```
##
## Note: only 18 parameters counted of 22 specified parameters
##
## AICc and parameter count have been adjusted upward
```

Ajuste dos modelos com heterogeneidade

Para os modelos com heterogeneidade acrescente o termo `mixture` nas fórmulas do parâmetro p:

```
## Fórmulas estatísticas para cada parâmetro do modelo com heterogeneidade
## p com heterogeneidade
t.mix <- list(formula=~mixture)
## p=c com heterogeneidade (use o argumento share=TRUE)
t.mixshared=list(formula=~mixture,share=TRUE)
## Parametros dependem do tempo
t.timemixshared <- list(formula=~time+mixture, share=TRUE)
t.timemix <- list(formula=~time+mixture)
```

E ajuste os modelos

```
boto.Mh <- mark(botoH, model.parameters=list(p=t.mixshared))
boto.Mbh <- mark(botoH, model.parameters=list(c=t.mix, p=t.mix))
boto.Mth <- mark(botoH, model.parameters=list(p=t.timeshared))
```

```
##
## Note: only 12 parameters counted of 13 specified parameters
##
## AICc and parameter count have been adjusted upward
```

```
boto.Mtbh <- mark(botoH, model.parameters=list(c=t.timemix, p=t.timemix))
```

```
##
## Note: only 22 parameters counted of 25 specified parameters
##
## AICc and parameter count have been adjusted upward
```

Seleção de modelos

A função abaixo retorna a tabela de seleção de modelos:

```
collect.models(lx=c("boto.M0", "boto.Mb", "boto.Mt", "boto.Mtb",
                    "boto.Mh", "boto.Mbh", "boto.Mth", "boto.Mtbh"))
```

```
## Warning in model.table(x, type, pf = 2, adjust = adjust): Model list contains models of differing ty
```

```
##                                                model npar     AICc DeltaAICc
## 8 pi(~1)p(~time + mixture)c(~time + mixture)f0(~1)   25 263.7916  0.000000
## 3                                  p(~time)c()f0(~1)   12 270.1989  6.407255
## 7                            pi(~1)p(~time)c()f0(~1)   13 272.3332  8.541585
## 6                 pi(~1)p(~mixture)c(~mixture)f0(~1)    6 275.2837 11.492097
## 5                         pi(~1)p(~mixture)c()f0(~1)    4 275.6888 11.897109
## 4                             p(~time)c(~time)f0(~1)   22 282.6256 18.833933
## 1                                     p(~1)c()f0(~1)    2 289.7349 25.943229
## 2                                   p(~1)c(~1)f0(~1)    3 290.1108 26.319200
##         weight Deviance
## 8 9.429729e-01 156.7423
## 3 3.829848e-02 191.7698
## 7 1.317400e-02 191.7698
## 6 3.013160e-03 209.4365
## 5 2.460791e-03 213.9520
## 4 7.669421e-05 182.3529
## 1 2.192797e-06 232.0679
## 2 1.817011e-06 230.4140
```

Valores das estimativas

A função `coef` retorna os coeficientes na escala de ligação (logito). Para as estimativas na escala de probabilidades use a função `get.real`:

```
coef(boto.Mtbh, data=boto2002)
```

```
##                    estimate           se           lcl          ucl
## pi:(Intercept)  -1.8514314    0.6056074    -3.0384219   -0.6644410
## p:(Intercept)    1.8253283    1.5984464    -1.3076268    4.9582834
## p:time2          0.3146987    0.6133085    -0.8873859    1.5167833
## p:time3          0.5438242    0.6672658    -0.7640168    1.8516651
## p:time4          0.9694277    0.7441941    -0.4891928    2.4280482
## p:time5         -0.4567758    1.1775055    -2.7646866    1.8511350
## p:time6          0.7470890    1.0099426    -1.2323986    2.7265766
## p:time7          0.4594252    1.2986955    -2.0860180    3.0048684
## p:time8        -14.7650900 2003.2167000 -3941.0698000 3911.5397000
## p:time9        -14.7650720 2067.4535000 -4066.9740000 4037.4438000
## p:time10         1.1526116    1.4787372    -1.7457134    4.0509365
## p:time11        25.8144280  852.0971900 -1644.2961000 1695.9250000
## p:mixture2      -2.9777979    1.6426965    -6.1974831    0.2418874
## c:(Intercept)    0.6693921    0.8386371    -0.9743366    2.3131209
## c:time3          0.1294258    0.9199078    -1.6735935    1.9324451
## c:time4          1.3456207    0.8546159    -0.3294265    3.0206679
## c:time5          1.4852605    0.8379450    -0.1571118    3.1276327
## c:time6          0.3168782    0.8548457    -1.3586194    1.9923759
## c:time7          1.0364551    0.8328759    -0.5959817    2.6688918
## c:time8         -1.8525306    1.0592575    -3.9286753    0.2236141
## c:time9         -0.6799929    0.9019384    -2.4477922    1.0878065
## c:time10        -0.9865895    0.9303093    -2.8099957    0.8368167
## c:time11        -0.7035456    0.9007602    -2.4690357    1.0619444
## c:mixture2      -2.3848798    0.4685488    -3.3032355   -1.4665241
## f0:(Intercept) -13.5769340  184.2390900  -374.6855600  347.5316900
```

```
## Na escala de probabilidades
get.real(boto.Mtbh, parameter="p")
```

```
## [[1]]
##                    1         2         3         4         5         6
## mixture:1 0.8612043 0.8947332 0.9144446 0.9423918 0.7971462 0.9290652
## mixture:2 0.2400383 0.3020045 0.3523683 0.4543669 0.1666934 0.4000203
##                    7            8           9        10 11
## mixture:1 0.9076064 2.400668e-06 2.40071e-06 0.9515675  1
## mixture:2 0.3333562 1.222058e-07 1.22208e-07 0.5000355  1
```

```
get.real(boto.Mtbh, parameter="c")
```

```
## [[1]]
##                    2         3         4         5         6         7
## mixture:1 0.6613670 0.6897216 0.8823643 0.8961027 0.7283506 0.8462969
## mixture:2 0.1524533 0.1699387 0.4085732 0.4426961 0.1980369 0.3364773
##                    8         9        10        11
## mixture:1 0.2344884 0.4973498 0.4213589 0.4914625
## mixture:2 0.0274376 0.0835180 0.0628509 0.0817328
```

```
get.real(boto.Mtbh, parameter="pi")
```

```
## [[1]]
##
## mixture:1 0.1357049
```