Intro to RMark - part 3

Philip Dixon

10/15/2021

Contents

This document describes a few additional features of RMark, then provides a cross-reference between the open population extensions discussed in lecture and the necessary RMark commands.

Additional useful features of RMark

more "automatic" variables (actually a feature of Mark) The intro document described how using time in a formula provided time-specific (e.g. year-specific) estimates for a parameter. There are other "automatic" variables:

- Time (note capitalization): use time as a regression variable with values 0, 1, 2. Fits a linear regression on the link scale, so logit(probability) = b0 + b1*time or log(count) = b0 + b1*time. The intercept is the predicted link-scale value in year 1, the first year of the data.

For example, one of the models fit in the Intro document was mT, p ~ Time.

```
reid.models[['mT']]$results$beta
```

```
##                    estimate           se           lcl          ucl
## p:(Intercept)   -0.2505940 2.196049e-01 -6.810196e-01 1.798315e-01
## p:Time           0.1873047 9.224140e-02  6.511400e-03 3.680979e-01
## f0:(Intercept) -19.8133970 1.326028e+04 -2.600996e+04 2.597034e+04
```

The estimated intercept and slope are -0.2505 and 0.187, so fitted equation is logit(capture probability) = -0.2505 + 0.187 * Time. For time 2, Time has the value of 1, so that predicted value is -0.2505 + 0.187*1 = -0.0635 and the backtransformed capture probability is 1/(1+exp(-(-0.0635))) = 0.484. You don't have to do the work, RMark provides that value as the real parameter for p at time 2:

```
reid.models[['mT']]$results$beta
```

```
##                    estimate           se           lcl          ucl
## p:(Intercept)   -0.2505940 2.196049e-01 -6.810196e-01 1.798315e-01
## p:Time           0.1873047 9.224140e-02  6.511400e-03 3.680979e-01
## f0:(Intercept) -19.8133970 1.326028e+04 -2.600996e+04 2.597034e+04
```

- Age: age of individual, as a regression variable. Default is to assume individuals are marked at age 0. You can provide initial ages, either for all individuals or for groups of individuals, e.g. newborn or adult (just like we specified captures histories for sex=Male and sex=Female). See initial.age = for how to specify initial ages.

To illustrate the difference between Time and Age, here are the values generated for newborns released in years 1, 2, and 3 and recaptured in years 2, 3, and 4:

Using Time:

Release	year 2	year 3	year 4
year 1	1	2	3
year 2		2	3
year 3			3

Using Age:

Release	year 2	year 3	year 4
year 1	1	2	3
year 2		1	2
year 3			1

You can combine age + time effects (an additive effects model for 2 regression variables). You can write the interaction between age and time, i.e. age:time to generate a different value for each cell of the table.

- age: Treats age as a factor variable (instead of a regression variable). Each number in the tables above corresponds to a unique mean (for each time or for each age).
- cohort: Generates a unique mean for each release occasion. A regression doesn't make sense here, so there is no Cohort. The numbers in the table below indicate each mean.

Using cohort:

Release	year 2	year 3	year 4
year 1	1	1	1
year 2		2	2
year 3			3

I use age and time a lot. I rarely have the need to use cohort.

Creating your own design matrix. This is possible but it requires a bit of work. You may remember that the mark() function runs 5 separate functions in order. To create your own design matrix, you need to intervene after the second of those separate functions. That means you have to run the 5 functions individually. The multi-year open population dipper data set has an example of doing this. These birds live along mountain streams. For those data, there was a flood in year 2 that was assumed to affect survival in years 2 and 3 and capture probability in year 3. ?dipper describes how to create and use these variables. Look for "Add Flood covariates" in the help file. The next few lines create covariate named Flood. Further down, you'll see formulae that include Flood.

Model averaging when some parameters are not identified

model.average() is very useful to model average a collection of models. By default, it model averages all parameters. Unfortunately, it is very conservative. If any parameter to be averaged is unidentified in a model (operationally defined by very large variance), that model is eliminated totally and completely. Even if you really only care about some other parameters that are identified in all models.

There are two solutions, depending on the relationship between the problem parameters and the ones you care about:

1) If they are different "types", e.g. one of the p's isn't identified but all you care about is f0 (for a closed population), then specify the parameter type you're interested in. For example, in the reid models from the intro part 1 document, the last p and f0 are not identified. You could model average c's by

```
model.average(reid.models, parameter='c')
```

```
##         par.index  estimate         se fixed    note group time Time c
## c g1 t2         7 0.6116022 0.06050248                   1    2    0 1
## c g1 t3         8 0.6010348 0.05903125                   1    3    1 1
## c g1 t4         9 0.6054390 0.05013745                   1    4    2 1
## c g1 t5        10 0.6139526 0.04805464                   1    5    3 1
## c g1 t6        11 0.6133941 0.04666496                   1    6    4 1
```

2) If they are not different types, e.g. in a general CJS model, the last survival and last capture probability are unidentified, but you want to estimate other survival parameters. To do this, you need to be more specific about what you want to model average. This requires finding the PIMS index number(s) for the parameters of interest. Pick any model and have RMark print out the PIMS index numbers:

```
reid.models[['mtb']]$pims
```

```
## $p
## $p[[1]]
## $p[[1]]$pim
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    2    3    4    5    6
##
## $p[[1]]$group
## [1] 1
##
##
##
## $c
## $c[[1]]
## $c[[1]]$pim
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    7    8    9   10   11
##
## $c[[1]]$group
## [1] 1
##
##
##
## $f0
## $f0[[1]]
## $f0[[1]]$pim
##      [,1]
## [1,]   12
##
## $f0[[1]]$group
## [1] 1
```

```
reid.models[['mt']]$pims
```

```
## $p
## $p[[1]]
## $p[[1]]$pim
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    2    3    4    5    6
##
## $p[[1]]$group
```

```
## [1] 1
##
##
##
## $c
## $c[[1]]
## $c[[1]]$pim
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    7    8    9   10   11
##
## $c[[1]]$group
## [1] 1
##
##
##
## $f0
## $f0[[1]]
## $f0[[1]]$pim
##      [,1]
## [1,]   12
##
## $f0[[1]]$group
## [1] 1
```

The choice of model doesn't matter. The index numbers are the same for all models. Most models force various values to be the same, but that doesn't matter. You see here that parameters 1 to 6 are the 6 initial capture probabilities, 7 to 11 are the 5 recapture probabilities and 12 is the # unseen.

Then tell RMark to model average specific parameters

```
model.average(reid.models, indices=c(1, 2, 7, 12))
```

```
##   par.index  estimate         se
## 1         1 0.3478323 0.07635712
## 2         2 0.3494347 0.08099008
## 3         7 0.6116022 0.06050248
## 4        12 2.7110810 3.18183066
```

will average p1, p2, c2 and f0.

Catalog of RMark model names

This is a simpler, condensed version of the MarkModels.pdf document that can be found on the class web site or downloaded in the RMark library folder.

Class model name	Mark model	parameters
Cormack-Jolly-Seber	CJS	Phi p
Pradel	PradSen	Phi p Gamma
Pradel with lambda	Pradlambda	Phi p Lambda
Seber Dead Recovery	Recovery	S r
Brownie Dead Recovery	Brownie	S f
Robust design	Robust	S p c gamma' gamma''
Multistate	Multistate	S p Psi
Superpopulation	POPAN	Phi p pent N

Notes: The MarkModels document says the Pradel with lambda model is PradLambda (capital L), but RMark wants Pradlambda. I didn't check PradSen. The Robust design model seems to deal with temporary emigration, since gamma' and gamma'' are parameters.