CJS Modeling in MRA

Trent McDonald

January 4, 2016

This vignette repeats and embellishes examples in `help(F.cjs.estim)`.

Time-varying Models

The following demonstrates two methods for fitting a time-varying capture and survival model, the so-called "small t" model. First, we attach the `mra` library and obtain access to the example dipper data.

```
library(mra)
```

```
## mra (version 2.16.11)
```

```
data("dipper.histories")
dim(dipper.histories)
```

```
## [1] 294   7
```

```
head(dipper.histories)
```

```
##   h1 h2 h3 h4 h5 h6 h7
## 1  1  1  1  1  1  1  0
## 2  1  1  1  1  1  0  0
## 3  1  1  1  1  0  0  0
## 4  1  1  1  1  0  0  0
## 5  1  1  0  1  1  1  0
## 6  1  1  0  0  0  0  0
```

Method 1: Using factors

The following code constructs a factor variable containing one level for each capture occasion. The attribute of this factor tells `mra` the "other"" dimension of the problem. Internally, `mra` will use this attribute to replicate the factor into matrices that are the appropriate size.

```
ct <- as.factor( paste("T",1:ncol(dipper.histories), sep=""))
attr(ct,"nan")<-nrow(dipper.histories)
ct
```

```
## [1] T1 T2 T3 T4 T5 T6 T7
## attr(,"nan")
## [1] 294
## Levels: T1 T2 T3 T4 T5 T6 T7
```

Next, call `F.cjs.estim` and specify that `ct` is a time-varying vector covariate using the `tvar` function. When the vector given to `tvar` is a factor, there are additional options which allow the user to drop certain levels of the factor. This is useful when coefficients for some levels are not estimable, as in the case of a completely time-varying CJS model. Here, there are 7 levels in factor `ct`, but only 6 capture and survival parameters are defined (recall, 1st capture parameter is not estimable, and 7th survival parameter between occasions 7 and 8 does not exist). Consequently, we tell `tvar` to drop the first two levels of `ct` from the capture model. We drop the level 1st because only 6 parameters exist. We drop the 2nd to break the colinearity of levels and

define p_2 as the reference level. In the survival model, we drop the first, sixth, and seventh levels of `ct`. The first level is dropped to break the colinearity of levels and define ϕ_1 as the reference level. We drop level 6 because the last survival and capture parameters are confounded in CJS models. We drop the 7th level because there are only 6 survival parameters. The call the `F.cjs.estim` is:

```
dipper1.cjs <- F.cjs.estim( ~tvar(ct,drop=c(1,2)), ~tvar(ct,drop=c(1,6,7)), dipper.histories )
dipper1.cjs
```

```
## Call:
## F.cjs.estim(capture = ~tvar(ct, drop = c(1, 2)), survival = ~tvar(ct,
##     drop = c(1, 6, 7)), histories = dipper.histories)
##
##  Capture var                  Est      SE          Survival var                      Est       SE
##  (Intercept)                  0.82928  0.78283     (Intercept)                       0.93546   0.7677
##  tvar(ct, drop = c(1, 2)):T3  1.65563  1.29086     tvar(ct, drop = c(1, 6, 7)):T2  -1.19828  0.8698
##  tvar(ct, drop = c(1, 2)):T4  1.5221   1.07234     tvar(ct, drop = c(1, 6, 7)):T3  -1.02284  0.8041
##  tvar(ct, drop = c(1, 2)):T5  1.37675  0.98779     tvar(ct, drop = c(1, 6, 7)):T4  -0.41986  0.8083
##  tvar(ct, drop = c(1, 2)):T6  1.79509  1.06789     tvar(ct, drop = c(1, 6, 7)):T5  -0.5361   0.8022
##  tvar(ct, drop = c(1, 2)):T7  0.2106   0.83736
##
## Message = SUCCESS: Convergence criterion met
## Link = logit
## Model df =  11
## Std Errors and QAIC adjusted for C_hat =  1 on 5 df
## Log likelihood =  -328.475105968236
## Deviance =  656.950211936473
## AIC =  678.950211936473
## AICc =  679.886382149239
## QAIC =  678.950211936473
## QAICc =  679.886382149239
##
## Population Size Estimates (se):
## N2=86 (21.39),  N3=84 (7.18),  N4=88 (6.29),  N5=98 (6.7),  N6=105 (5.85),  N7=126 (30.49),
```

Method 2: Using explicit 2-D matricies

While using factors (*Method 1* above) produces the most economical code, it does not adequately illuminate the covariate matrices which are at the heart of CJS modeling. To illustrate covariates as explicit matricies, this method constructs one 2-D matrix for each paramter, then estimates the same model as *Method 1*.

First, we construct 6 matricies containing 1's in a single column only. In *Method 1*, this construction was performed behind-the-scenes by `tvar`. Note that only 6 matricies are required due to the number of parameters, breaking of colinearity, and confounding of CJS paramters mentioned above.

```
x2 <- matrix(c(0,1,0,0,0,0,0), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
x3 <- matrix(c(0,0,1,0,0,0,0), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
x4 <- matrix(c(0,0,0,1,0,0,0), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
x5 <- matrix(c(0,0,0,0,1,0,0), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
x6 <- matrix(c(0,0,0,0,0,1,0), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
x7 <- matrix(c(0,0,0,0,0,0,1), nrow(dipper.histories), ncol(dipper.histories), byrow=TRUE)
```

Each of the above matrices have a column of 1's corresponding to the effect they estimate. The first six rows of `x3` and `x4` are:

```
head(x3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    0    0    1    0    0    0    0
## [2,]    0    0    1    0    0    0    0
## [3,]    0    0    1    0    0    0    0
## [4,]    0    0    1    0    0    0    0
## [5,]    0    0    1    0    0    0    0
## [6,]    0    0    1    0    0    0    0
```

```
head(x4)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    0    0    0    1    0    0    0
## [2,]    0    0    0    1    0    0    0
## [3,]    0    0    0    1    0    0    0
## [4,]    0    0    0    1    0    0    0
## [5,]    0    0    0    1    0    0    0
## [6,]    0    0    0    1    0    0    0
```

We now call `F.cjs.extim` without aid of `tvar` by explicitely specifying the matricies in each model. Note that `x2` is not included in the capture model, and that `x6` and `x7` are not included in the survival model.

```
dipper2.cjs <- F.cjs.estim( ~x3+x4+x5+x6+x7, ~x2+x3+x4+x5, dipper.histories )
dipper2.cjs
```

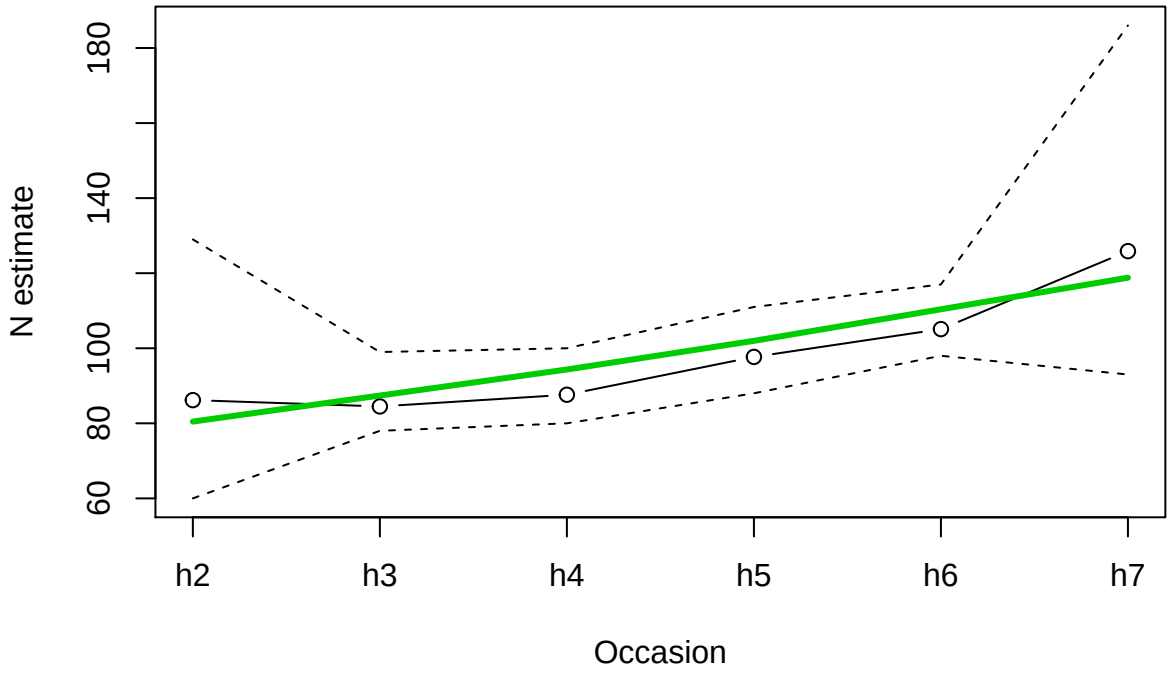
```
## Call:
## F.cjs.estim(capture = ~x3 + x4 + x5 + x6 + x7, survival = ~x2 +
##     x3 + x4 + x5, histories = dipper.histories)
##
##  Capture var   Est      SE           Survival var   Est       SE
##  (Intercept)   0.82928  0.78283      (Intercept)    0.93546   0.76772
##  x3            1.65563  1.29086      x2             -1.19828  0.8698
##  x4            1.5221   1.07234      x3             -1.02284  0.80412
##  x5            1.37675  0.98779      x4             -0.41986  0.80834
##  x6            1.79509  1.06789      x5             -0.5361   0.80229
##  x7            0.2106   0.83736
##
## Message = SUCCESS: Convergence criterion met
## Link = logit
## Model df =  11
## Std Errors and QAIC adjusted for C_hat =  1 on 5 df
## Log likelihood =  -328.475105968236
## Deviance =  656.950211936473
## AIC =  678.950211936473
## AICc =  679.886382149239
## QAIC =  678.950211936473
## QAICc =  679.886382149239
##
## Population Size Estimates (se):
## N2=86 (21.39),  N3=84 (7.18),  N4=88 (6.29),  N5=98 (6.7),  N6=105 (5.85),  N7=126 (30.49),
```

Note that parameter estimates produced by *Method 1* and *Method 2* are identical.

Plot: $\hat{N}_j$ estimates

Following is a plot of the Horvitz-Thomson population size estimates.

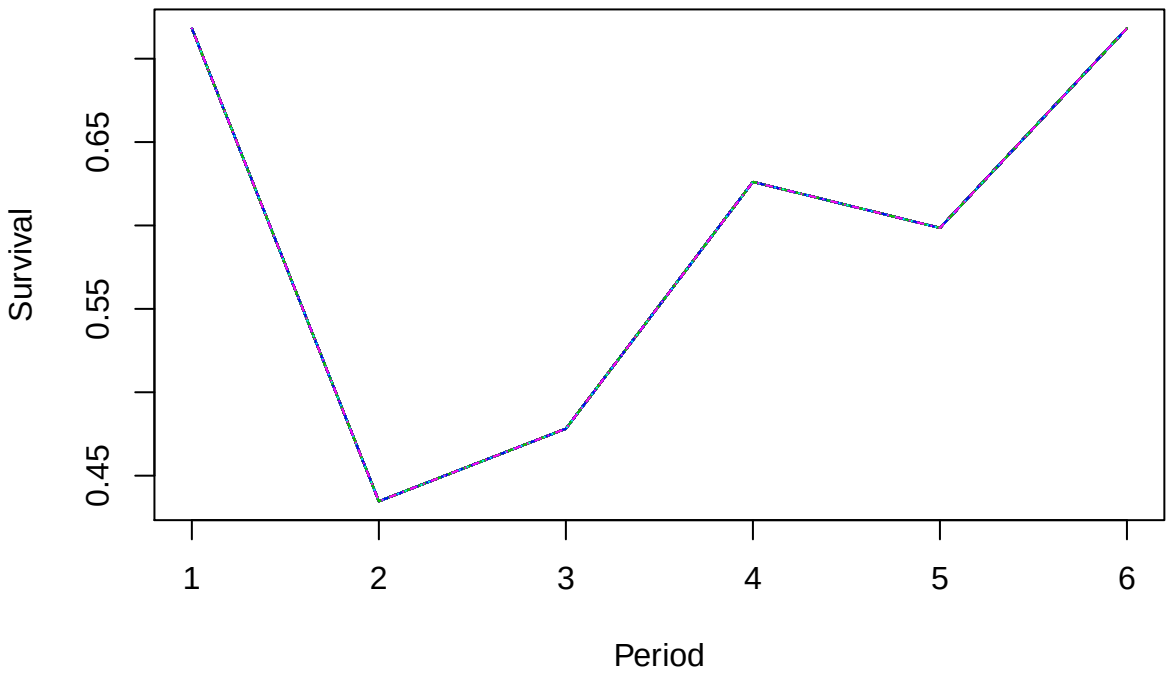
```
plot(dipper1.cjs)
```



Plot: $\hat{\phi}_j$ estimates

Following is a plot of survival estimates containing one line per individual.

```
plot(dipper1.cjs,type="s",ci=FALSE)
```


Survival
0.45
0.55
0.65
1
2
3
4
5
6
Period