Linear Models and Capture-Recapture Analysis in MARK

Jeff Laake

May 2, 2011

Analysis of capture-recapture data is somewhat different than what you you may have learned in your statistics courses but it also has some very strong similarities because it is based on linear models. In fact, much of statistical analysis is based on linear models because linear functions are both easy to understand and to manipulate. Capture-recapture models are linear models with a twist! To illustrate this and to help you see how it is related to what you have seen and learned before, I'll provide several examples that start with linear regression and then move to a simple capture-recapture model. In doing so, I'll show the algebraic representation of the model, provide some example data and then show the results of the analysis. Hopefully, this well help set a context for the remainder of the RMark workshop.

Let's start with a muliple linear regression model for weight as a function of length and sex of some critter. Generically, we can specify the linear prediction equation as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

where y is the dependent variable (e.g., weight),β_0 is the intercept and β_1 and β_2 are the parameters for $x_1$and $x_2$which are the independent predictor variables length and sex. A statistical model for the observed data (y_i, x_{1i}, x_{2i}) could be expressed as:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \varepsilon_i$$

where $\varepsilon_i \sim N(0, \sigma^2)$. Below are some sample data:

Table 1: Data in dataframe morphdata for multiple linear regression example to predict weight from length and sex.

	Weight	Length	Sex
1	60.570	11.328	Male
2	61.279	11.861	Female
3	69.737	12.864	Male
4	75.937	14.541	Female
5	59.933	11.008	Male
6	75.838	14.492	Female
7	78.790	14.723	Male
8	68.632	13.304	Female
9	66.744	13.146	Male
10	54.474	10.309	Female
11	59.065	11.030	Male
12	56.180	10.883	Female
13	68.969	13.435	Male
14	60.885	11.921	Female
15	73.873	13.849	Male
16	66.481	12.488	Female
17	71.786	13.588	Male
18	77.379	14.960	Female
19	63.420	11.900	Male
20	69.371	13.887	Female
21	76.745	14.674	Male
22	56.712	11.061	Female
23	70.203	13.258	Male
24	56.789	10.628	Female
25	61.825	11.336	Male
26	61.406	11.931	Female
27	53.955	10.067	Male
28	62.605	11.912	Female
29	76.577	14.348	Male
30	59.476	11.702	Female

In R this linear model with a normal error distribution can be easily fitted with the function lm. The code and results are shown below:

```
> WeightModel = lm(Weight ~ Length + Sex, data = morphdata)
> summary(WeightModel)

Call:
lm(formula = Weight ~ Length + Sex, data = morphdata)

Residuals:
     Min       1Q   Median       3Q      Max
-2.94372 -0.51634 -0.04649  0.91268  1.76804

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.3133     1.9027   1.216 0.234594
Length        4.9965     0.1515  32.979  < 2e-16 ***
SexMale       1.6920     0.4398   3.847 0.000663 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.198 on 27 degrees of freedom
Multiple R-squared: 0.9769,        Adjusted R-squared: 0.9752
F-statistic: 571.4 on 2 and 27 DF,  p-value: < 2.2e-16
```

The particulars of the results are not that important for what I want to show. What does matter is how the formula relates to the linear equation shown above. We can see the values of the predictor variables for this model by using a function in R called model.matrix. The lm function handles all of this internally but to understand the link between regression and capture-recapture it is important to understand what is going on under the hood. The model.matrix function provides the design matrix which is a matrix of the predictor variables with a row for each observation and a column for each of the predictor variables (e.g., variables in the algebraic equation). The code to create the design matrix for this model and data is:

```
> dm = model.matrix(~Length + Sex, data = morphdata)
```

Table 2: Design matrix for predictor variables for model of weight from length and sex (Length+Sex).

	(Intercept)	Length	SexMale
1	1.000	11.328	1.000
2	1.000	11.861	0.000
3	1.000	12.864	1.000
4	1.000	14.541	0.000
5	1.000	11.008	1.000
6	1.000	14.492	0.000
7	1.000	14.723	1.000
8	1.000	13.304	0.000
9	1.000	13.146	1.000
10	1.000	10.309	0.000
11	1.000	11.030	1.000
12	1.000	10.883	0.000
13	1.000	13.435	1.000
14	1.000	11.921	0.000
15	1.000	13.849	1.000
16	1.000	12.488	0.000
17	1.000	13.588	1.000
18	1.000	14.960	0.000
19	1.000	11.900	1.000
20	1.000	13.887	0.000
21	1.000	14.674	1.000
22	1.000	11.061	0.000
23	1.000	13.258	1.000
24	1.000	10.628	0.000
25	1.000	11.336	1.000
26	1.000	11.931	0.000
27	1.000	10.067	1.000
28	1.000	11.912	0.000
29	1.000	14.348	1.000
30	1.000	11.702	0.000

The predicted weight values can be computed by matrix multiplication using the design matrix and the estimated coefficients. The code to compute the predicted values is:

```
> pred.wt = dm %*% coef(WeightModel)
```

The predicted values and the set of linear equations are shown below. The intercept for females is β_0 and $\beta_0 + \beta_2$ for males:

```
 [1] 60.604 = 2.313*1 + 4.997*11.328 + 1.692*1
 [2] 61.575 = 2.313*1 + 4.997*11.861 + 1.692*0
 [3] 68.282 = 2.313*1 + 4.997*12.864 + 1.692*1
 [4] 74.968 = 2.313*1 + 4.997*14.541 + 1.692*0
 [5] 59.009 = 2.313*1 + 4.997*11.008 + 1.692*1
 [6] 74.723 = 2.313*1 + 4.997*14.492 + 1.692*0
 [7] 77.571 = 2.313*1 + 4.997*14.723 + 1.692*1
 [8] 68.787 = 2.313*1 + 4.997*13.304 + 1.692*0
 [9] 69.688 = 2.313*1 + 4.997*13.146 + 1.692*1
[10] 53.822 = 2.313*1 + 4.997*10.309 + 1.692*0
[11] 59.116 = 2.313*1 + 4.997*11.030 + 1.692*1
[12] 56.689 = 2.313*1 + 4.997*10.883 + 1.692*0
[13] 71.134 = 2.313*1 + 4.997*13.435 + 1.692*1
[14] 61.875 = 2.313*1 + 4.997*11.921 + 1.692*0
[15] 73.203 = 2.313*1 + 4.997*13.849 + 1.692*1
[16] 64.712 = 2.313*1 + 4.997*12.488 + 1.692*0
[17] 71.899 = 2.313*1 + 4.997*13.588 + 1.692*1
[18] 77.059 = 2.313*1 + 4.997*14.960 + 1.692*0
[19] 63.465 = 2.313*1 + 4.997*11.900 + 1.692*1
[20] 71.701 = 2.313*1 + 4.997*13.887 + 1.692*0
[21] 77.322 = 2.313*1 + 4.997*14.674 + 1.692*1
[22] 57.578 = 2.313*1 + 4.997*11.061 + 1.692*0
[23] 70.251 = 2.313*1 + 4.997*13.258 + 1.692*1
[24] 55.415 = 2.313*1 + 4.997*10.628 + 1.692*0
[25] 60.646 = 2.313*1 + 4.997*11.336 + 1.692*1
[26] 61.925 = 2.313*1 + 4.997*11.931 + 1.692*0
[27] 54.305 = 2.313*1 + 4.997*10.067 + 1.692*1
[28] 61.832 = 2.313*1 + 4.997*11.912 + 1.692*0
[29] 75.698 = 2.313*1 + 4.997*14.348 + 1.692*1
[30] 60.781 = 2.313*1 + 4.997*11.702 + 1.692*0
```

In the above model, only the intercept varied for the sexes. Now, let's consider a model in which the intercept and the slope for length varies by sex. Algebraically, this is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

The interaction term is a multiplication of the two predictor variables Length and Sex. The Length slope for females will be β_1 and for males it will be $\beta_1 + \beta_3$. The formula is the multiplication (in formula speak) of Length and Sex (Length*Sex).

```
> dm = model.matrix(~Length * Sex, data = morphdata)
```

Table 3: Design matrix for predictor variables for model of weight from length and sex (Length*Sex).

	(Intercept)	Length	SexMale	Length:SexMale
1	1.000	11.328	1.000	11.328
2	1.000	11.861	0.000	0.000
3	1.000	12.864	1.000	12.864
4	1.000	14.541	0.000	0.000
5	1.000	11.008	1.000	11.008
6	1.000	14.492	0.000	0.000
7	1.000	14.723	1.000	14.723
8	1.000	13.304	0.000	0.000
9	1.000	13.146	1.000	13.146
10	1.000	10.309	0.000	0.000
11	1.000	11.030	1.000	11.030
12	1.000	10.883	0.000	0.000
13	1.000	13.435	1.000	13.435
14	1.000	11.921	0.000	0.000
15	1.000	13.849	1.000	13.849
16	1.000	12.488	0.000	0.000
17	1.000	13.588	1.000	13.588
18	1.000	14.960	0.000	0.000
19	1.000	11.900	1.000	11.900
20	1.000	13.887	0.000	0.000
21	1.000	14.674	1.000	14.674
22	1.000	11.061	0.000	0.000
23	1.000	13.258	1.000	13.258
24	1.000	10.628	0.000	0.000
25	1.000	11.336	1.000	11.336
26	1.000	11.931	0.000	0.000
27	1.000	10.067	1.000	10.067
28	1.000	11.912	0.000	0.000
29	1.000	14.348	1.000	14.348
30	1.000	11.702	0.000	0.000

Now let's switch to an example that is more closely aligned to capture-recapture. Instead of a continuous dependent variable like weight, we'll consider a Bernoulli random variable with values 0 or 1 for failure/success or in this case death/survival. The prediction equation for the weight regression above is for the mean (E(y)) because the error is assumed to have 0 mean and variance σ^2. For a Bernoulli random variable the "mean" (E(y)) is p, which is the probability of success (survival in this case). Now the prediction equation is for p and not y=0 or 1 but in both cases it is for the expected value of the distribution. I'll use the same morphdata and simulate a Bernoulli random variable with p being a function of a condition index (weight/length) and sex. Now because p is bounded between 0 and 1, we can't simply have a linear function of the predictor variables for p because it could have non-admissable values. The solution is a link function that relates the covariates to the parameter p. A standard link function is the logit (logistic regression). For this example, the logit is expressed as follows:

$$ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

where $x_1$and x_2 are the independent predictor variables condition=weight/length and sex. The inverse link function provides an expression for p in terms of the covariates:

$$p = \frac{exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}{1 + exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)} = \frac{1}{1 + exp(-\beta_0 - \beta_1 x_1 - \beta_2 x_2)}.$$

In R, the inverse logit function is easily calculated with the plogis function. It uses the second formulation above because it is numerically more stable for large values of the exponential. In MARK-speak, p is a real parameter and β_0, β_1andβ_2 are the beta parameters. The beta parameters can take any value and the real parameter in this case is constrained between 0-1. Expressed in terms of the logit, we still have a linear model but p is clearly not a linear function of the covariates. Always important to remember this! Models for Bernoulli (or Binomial = sums of Bernoullis with same p) random variables can be fitted with the function glm (generalized linear model) by specifying the family argument of glm. Below is an example that shows how the data are simulated and then a model is fitted for the simulated data:

```
> morphdata$CI = scale(morphdata$Weight/morphdata$Length)
> morphdata = rbind(morphdata, morphdata, morphdata)
> p = plogis(-1 + 2 * morphdata$CI - 0.4 * (as.numeric(morphdata$Sex) -
+     1))
> morphdata$Alive = rbinom(nrow(morphdata), 1, p)
> head(morphdata)
```

```
    Weight   Length    Sex         CI Alive
1 60.57032 11.32754   Male  0.8145098     1
2 61.27881 11.86062 Female -0.7169562     0
3 69.73709 12.86427   Male  1.4405214     1
4 75.93703 14.54104 Female -0.2447954     1
5 59.93290 11.00841   Male  1.6380345     1
6 75.83821 14.49195 Female -0.1526057     1
```

```
> SurvivalModel = glm(Alive ~ CI + Sex, family = binomial, data = morphdata)
> summary(SurvivalModel)

Call:
glm(formula = Alive ~ CI + Sex, family = binomial, data = morphdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5986  -0.6508  -0.1929   0.6670   2.7499

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.6253     0.4557  -1.372    0.170
CI            2.5741     0.6119   4.206 2.59e-05 ***
SexMale      -1.0256     0.7546  -1.359    0.174
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 118.288  on 89  degrees of freedom
Residual deviance:  74.154  on 87  degrees of freedom
AIC: 80.154

Number of Fisher Scoring iterations: 6
```

The true beta values are -1,2, and -0.4. The first 6 rows of the design matrix for the logits is:

```
> dm = model.matrix(~CI + Sex, morphdata)
> head(dm)

  (Intercept)         CI SexMale
1           1  0.8145098       1
2           1 -0.7169562       0
3           1  1.4405214       1
4           1 -0.2447954       0
5           1  1.6380345       1
6           1 -0.1526057       0
```

and the predicted values can be computed as follows:

```
> beta = coef(SurvivalModel)
> Ey = plogis(dm %*% beta)
> head(Ey)
```

```
        [,1]
1 0.60960984
2 0.07792914
3 0.88666084
4 0.22175966
5 0.92860748
6 0.26539037
```

with the equations expressed as:

```
[1] 0.6096 = 1/(1+exp(0.62532*1 + -2.574*0.81451 + 1.0256*1)
[2] 0.07793 = 1/(1+exp(0.62532*1 + -2.574*-0.71696 + 1.0256*0)
[3] 0.8867 = 1/(1+exp(0.62532*1 + -2.574*1.4405 + 1.0256*1)
[4] 0.2218 = 1/(1+exp(0.62532*1 + -2.574*-0.24480 + 1.0256*0)
[5] 0.9286 = 1/(1+exp(0.62532*1 + -2.574*1.6380 + 1.0256*1)
[6] 0.2654 = 1/(1+exp(0.62532*1 + -2.574*-0.15261 + 1.0256*0)
```

The above example would be equivalent to a known-fate analysis with a single interval in which survival could take place. That is a fairly simple and unrealistic example. To make it more realistic and start to introduce the concepts used in MARK, I'll expand the example to include two intervals. Each of the animals is released at the initiation of the study and it is observed at time 1 and then at time 2 and a 0 is recorded if it died and a 1 if it survived. The possible values are 11, 10, and 00. The history 01 is not allowed because it is known fate and once it dies in the first time, it is obviously dead at the second time. The probabilities for these capture histories are p^2, $p(1-p)$ and $(1-p)$, respectively. If we allowed survival to differ for the 2 intervals, then we would express the probabilities as: $p_1 p_2$, $p_1(1-p_2)$ and $(1-p_1)$. We can no longer use either lm or standard glm to fit models to these data. However, it is relatively easy to program the likelihood and solve it numerically for the parameters. What we will focus on here is the model for the parameters in terms of the covariates.

Let's start with a fairly simple example in which we assume that the parameters depend only on time. A model for the logit of p might be as follows.

$$\begin{aligned} ln\left(\frac{p_1}{1-p_1}\right) &= \beta_0 \\ ln\left(\frac{p_2}{1-p_2}\right) &= \beta_0 + \beta_1 \end{aligned}$$

and another representation using a design matrix would be:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

Now what if we wanted to split the sexes into 2 groups to fit separate models. We might consider using 4 indices to represent the parameters with 1&2 for females times 1 and 2 and 3&4 for males times 1 and 2. In that case, we can represent the model with 4 logits,

but we'll still assume no differences between the sexes. Then the model is:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \\ ln\left(\frac{p_3}{1-p_3}\right) \\ ln\left(\frac{p_4}{1-p_4}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

If we want to have an additive sex difference the model might look like:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \\ ln\left(\frac{p_3}{1-p_3}\right) \\ ln\left(\frac{p_4}{1-p_4}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

If we want to have a different temporal pattern for each sex then the model would be:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \\ ln\left(\frac{p_3}{1-p_3}\right) \\ ln\left(\frac{p_4}{1-p_4}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}.$$

Notice that the last column in the design matrix is the product of the second and third columns because it is the interaction of sex and time. I can generate those design matrices with model.matrix and a formula if I create some data that represent the parameters. Below I create some data for the 4 parameter structure and show the formula and resulting design matrices.

```
> design.data = data.frame(parameter = 1:4, time = c("1", "2",
+     "1", "2"), sex = c("Female", "Female", "Male", "Male"))
> design.data

  parameter time    sex
1         1    1 Female
2         2    2 Female
3         3    1   Male
4         4    2   Male

> model.matrix(~time, design.data)

  (Intercept) time2
1           1     0
2           1     1
3           1     0
4           1     1
attr(,"assign")
```

```
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"
```

```
> model.matrix(~time + sex, design.data)
```

```
  (Intercept) time2 sexMale
1           1     0       0
2           1     1       0
3           1     0       1
4           1     1       1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"

attr(,"contrasts")$sex
[1] "contr.treatment"
```

```
> model.matrix(~time * sex, design.data)
```

```
  (Intercept) time2 sexMale time2:sexMale
1           1     0       0             0
2           1     1       0             0
3           1     0       1             0
4           1     1       1             1
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"

attr(,"contrasts")$sex
[1] "contr.treatment"
```

```
> model.matrix(~-1 + time:sex, design.data)
```

```
  time1:sexFemale time2:sexFemale time1:sexMale time2:sexMale
1               1               0             0             0
2               0               1             0             0
3               0               0             1             0
4               0               0             0             1
attr(,"assign")
[1] 1 1 1 1
```

```
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"

attr(,"contrasts")$sex
[1] "contr.treatment"
```

varies by time for males but not for females. It might look something like:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \\ ln\left(\frac{p_3}{1-p_3}\right) \\ ln\left(\frac{p_4}{1-p_4}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

The intercept (first column) is the value for female survival and second and third columns are the added values for time 1 and 2 for males. How could you construct a formula to create that design matrix. One way would be to create a dummy variable that is 0 for females and 1 for males and interact the male dummy variable with time as shown below:

```
> design.data$male = ifelse(design.data$sex == "Male", 1, 0)
> design.data

  parameter time    sex male
1         1    1 Female    0
2         2    2 Female    0
3         3    1   Male    1
4         4    2   Male    1

> model.matrix(~male:time, design.data)

  (Intercept) male:time1 male:time2
1           1          0          0
2           1          0          0
3           1          1          0
4           1          0          1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"
```

differs for all of the individuals in the data. The reason is that it does not fit into this structure because I don't have a $p_1$and a p_2 for each female and a $p_3$and a $p_4$for each male.

If I wanted to include CI with the above model, what I could do is to create a design matrix that looks as follows:

$$\begin{bmatrix} ln\left(\frac{p_1}{1-p_1}\right) \\ ln\left(\frac{p_2}{1-p_2}\right) \\ ln\left(\frac{p_3}{1-p_3}\right) \\ ln\left(\frac{p_4}{1-p_4}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & CI \\ 1 & 0 & 0 & CI \\ 1 & 1 & 0 & CI \\ 1 & 0 & 1 & CI \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}.$$

In computing the parameters for each animal, the value of CI for that animal would be plugged into the location for CI to give the design matrix and parameters for that animal.

This last example illustrates the concepts of PIMS and design matrices used in MARK. The parameters are indexed with the numbers 1,2,3,4 for the 4 parameters. With 4 real parameters, there are 4 rows in the design matrix. The number of columns (number of betas) depends on the model and varied from 2 to 4 above. The creation of the design data and the use of formula is how RMark works to specify the models for MARK.