

MA888 Ecological Statistics

Program Mark and RMark: Capture-recapture models

R.S.McCrea and B.J.T.Morgan

Spring Term 2012

1 Introduction

The development of computer software capable of fitting MRR models has been a catalyst in the research area and has sparked a wide range of applications of the models to numerous data sets. **Program Mark**¹ was developed to provide a common programming environment for the estimation of survival from marked animals. Prior to Mark no program easily combined the estimation of survival from both live and dead encounters, nor allowed for the modelling of capture and recapture probabilities in a general modelling framework for estimation of population size in closed populations.

The **RMark**² library is a collection of R functions that can be used as an interface to MARK for analysis of capture-recapture data. The library contains various functions that import/export capture data, build capture-recapture models, run the FORTRAN program MARK.EXE, and extract and display output. Program MARK has its own user interface; however, model development can be rather time-consuming and error-prone because the parameter structure and design matrix are created by hand. This interface in R was created to use the formula and design matrix functions in R to ease model development and reduce errors.

2 RMark

2.1 Installing RMark

1. Download and unzip the Program Mark folder from the MA888 module on moodle into your bodiam directory Z:\.
2. Create an empty directory called **MarkExample**.

¹White, G.C. and Burnham, K.P. (1999), Program Mark: survival estimation from populations of marked animals, *Bird Study* **46**, S120–S138

²Laake, J. and Rexstad, E. (2008)

3. Start up R. From within R, select **File** and choose to **Change dir....** Navigate to your new empty directory.
4. From within R, select **Packages** from the menu and choose **Install package(s) from local zip** at the bottom of the list.
5. Doing so will show a **select files** window. Navigate to **Z:** and select the **RMark.zip** file.
6. Issue the command `library(RMark)` within the R command window.
7. Next issue the command `MarkPath="Z:/RMark/Mark/"`.
8. The initial set up is now complete.

It is impossible to teach everything that **Mark** is capable of in one computing session. The instructions are written in a way that should explain what you are doing, rather than just giving you some computer code to copy. You will be shown how to fit time/group-dependencies and how to rank the fitted models by AIC value. Unlike Program **Mark**, in **RMark** there is no need to enter the individual parameter index matrices (PIMs) for each parameter, as the R functions are capable of constructing these from appropriate commands. This makes **RMark** a much more efficient way of fitting CJS-type models.

2.2 Cormorant example

This data set is currently in `*.inp` format, which is the format required for the windows version of **Mark**. In order to read it into **RMark** it needs to be converted. Follow the start-up instructions for **RMark** and once you have installed the **RMark** library read in the data:

```
> cormorants=convert.inp("Z:/RMark/cormorants.inp")
```

1. Before a model is fitted to the data, the data need to be processed and important attributes need to be defined such as:
 - (a) `model`: the type of analysis model (e.g., 'CJS', 'Known', 'POPAN')
 - (b) `begin.time`: the time of the first capture/release occasion for labelling
 - (c) `time.intervals`: the lengths of the time intervals between capture occasions
 - (d) `groups`: the list of factor variables in the data to define groups
 - (e) `initial.ages`: the age of animals at first capture/release corresponding to the levels of the age grouping variable (`age.var`)
 - (f) `nocc`: number of capture/encounter occasions which is determined from the contents of the 'ch' field in the data and the type of analysis model(`model`).

For this example, we are going to be fitting the CJS model and the data were collected from 1983 onwards. Therefore, to process the data type:

```
> corm.process=process.data(cormorants,model="CJS",
  begin.time=1983)
```

2. The next step is to instruct the program to create definitions of parameters which depends on the model you selected in the previous step:

```
> corm.ddl=make.design.data(corm.process)
```

3. Now we want to fit some models. Following along the lines of `Mark`, a model is described by sub-models for each parameter of the particular type of mark-recapture analysis. With the `corm` data we are using the CJS model with parameters ϕ and p . A parameter specification (sub-model) is defined by a list, although in most circumstances the list will only contain a single element, named the formula. For reasons that will be obvious later, the parameter specifications should be assigned to an object named with a prefix, this being the parameter name and the suffix being a description for the formula. For example, with the simple model, with constant survival and capture probability the specifications would be:

```
> Phi.dot=list(formula=~1)
```

```
> p.dot=list(formula=~1)
```

4. Create more parameter specifications:

```
> Phi.time=list(formula=~time)
```

```
> p.time=list(formula=~time)
```

5. Now we finally fit the models. To keep all of the models well defined, one can use the data name and add on the parameter specifications as in the following examples:

```
> corm.phi.dot.p.dot=mark(corm.process,
  corm.ddl,model.parameters=list(Phi=Phi.dot,p=p.dot))
```

```
> corm.phi.time.p.dot=mark(corm.process,
  corm.ddl,model.parameters=list(Phi=Phi.time,p=p.dot))
```

```
> corm.phi.time.p.time=mark(corm.process,
  corm.ddl,model.parameters=list(Phi=Phi.time,p=p.time))
```

```
> corm.phi.dot.p.time=mark(corm.process,
  corm.ddl,model.parameters=list(Phi=Phi.dot,p=p.time))
```

To see the output from any of these models you can use the command:

```
> summary(NAME OF MODEL)
```

Look at the output for the model `corm.phi.dot.p.dot`.

6. Look at the output for the model `corm.phi.time.p.dot`. Do survival probabilities appear to differ over time?
7. By looking at the output from the two nested models `corm.phi.time.p.dot` and `corm.phi.dot.p.dot`, conduct a likelihood-ratio test to see if the survival probabilities differ significantly over time.
8. To compare the results from all of the models you have fitted you need to collect them together:

```
> corm.cjs.results=collect.models()
```

9. Display the list of all models fitted:

```
> corm.cjs.results
```

10. The models in this list are ranked by AICc. Out of the 4 models you have fitted, which is the most appropriate for the data?

NOTE: Keep R open so that you can compare the RMark output with the Program Mark output.

3 Program Mark

Program Mark is the original software capable of fitting capture-recapture models and has a windows interface. The software is located in the SMSAS Departmental Software directory. Open Program Mark 5.0.

3.1 Analysis of Cormorant Data

1. Select **File** and select **New**. A window opens which allows you to load the data and select the type of data you wish to analyse.
2. Add title **Cormorant** to the **Title for this set of data** box and press **Click to select File**. Navigate your way to your Mark directory and select the file `cormorants.inp`. If you click **View File** you will be able to see the encounter history input.
3. This data set has 11 encounter occasions, and you need to select the **Recaptures only** input option on the left hand bar. All other options can be left as default.
4. Click **OK**.
5. We now wish to fit the four models we previously fitted in RMark. To do this, select **Run** and select **Pre-defined Model(s)**.

6. Add a title for the analyses.
7. Click **Select Models**. On the **Phi** tab press **Select All** and similarly on the **p** tab.
8. Press **OK** to return to the **Setup** screen.
9. Leave all other options on the default setting and click **OK to run**.
10. A table of results is now shown. You have fitted the same models as you fitted using **RMark**. Models are ranked using the information criterion **AICc**.
11. You should be able to see a difference in ordering of the models from **RMark** and **Mark**. Can you see which values in the table are different? The reason for this difference is to do with a statistical property called *parameter redundancy*. We will discuss this phenomenon in detail later in the course and we will run a computer practical using software **Maple** which is capable of detecting this feature. The correct model order is in fact that produced by **Mark**; what conclusions can you draw about the best model for the data?
12. You can look at the parameter estimates by right clicking on the model listed in the table.
13. Finally, let us look at a summary m-array of the input data. Select the **Output** list and select **Input data summary**. This will show you the data presented in an m-array like those shown during the lecture course. Comment on the off-diagonal m-array entries.

MA888: Ecological Statistics Assessment 3

Capture-recapture models: RMark

Please hand in your assessment to the SMSAS general office

by 12.00 on Friday 30th March 2012.

Question 1: Analysing dipper data

The dipper data and a number of other data sets are already contained in the RMark package. The aim of this example is to fit a Cormack-Jolly-Seber model to a data set. You will be shown how to fit time/group-dependencies and how to rank the fitted models by AIC value. The dipper data has groupings by sex, and therefore we can fit models with survival and capture probabilities which are different between the sexes.

- (a) Extract the data from the RMark package and put a copy of it in your directory:

```
> data(dipper)
```

- (b) Check this is the only contents of your workspace

```
> ls()
```

- (c) Obtain a summary of the dipper data

```
> summary(dipper)
```

- (d) How many female birds are in the data set?

- (e) View the first 5 records of the dipper data

```
> dipper[1:5,]
```

For this example, we are going to be fitting the CJS model, the data were collected from 1980 onwards, and the data are grouped into males and females. Therefore, to process the data type:

```
> dipper.process=process.data(dipper,model="CJS",  
  begin.time=1980,groups="sex")
```

- (f) The next step is to instruct the program to create the PIM structure which depends on the model you selected in the previous step:

```
> dipper.ddl=make.design.data(dipper.process)
```

- (g) Now we want to fit some models. Following along the lines of **Mark**, a model is described by sub-models for each parameter of the particular type of mark-recapture analysis. With the dipper data, we are using the CJS model with parameters ϕ and p . A parameter specification (sub-model) is defined by a list. For reasons that will be obvious later, the parameter specifications should be assigned to an object named with a prefix being the parameter name and the suffix being a description for the formula or some other strategy like numbering. For example, with the simple model, with constant survival and capture probability the specifications would be:

```
> Phi.dot=list(formula=~1)
> p.dot=list(formula=~1)
```

- (h) Create more parameter specifications:

```
> Phi.time=list(formula=~time)
> Phi.sex=list(formula=~sex)
> Phi.sexplusage=list(formula=~sex+age)
> p.time=list(formula=~time)
> p.Time=list(formula=~Time)
> p.Timeplussex=list(formula=~Time+sex)
```

- (i) Now we finally fit the models. To keep all of the models well defined, one way is to use the data name and add on the parameter specifications as in the following examples:

```
> dipper.phi.dot.p.dot=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.dot,p=p.dot))
> dipper.phi.time.p.dot=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.time,p=p.dot))
> dipper.phi.sex.p.dot=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.sex,p=p.dot))
> dipper.phi.sex.p.Timeplussex=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.sex,p=p.Timeplussex))
> dipper.phi.time.p.time=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.time,p=p.time))
> dipper.phi.sexplusage.p.dot=mark(dipper.process,
  dipper.ddl,model.parameters=list(Phi=Phi.sexplusage,p=p.dot))
```

Six of the possible sixteen models are listed above. Using an appropriate naming scheme, fit all 16 models to the dipper data set. To see the output from any of these models you can use the command:

```
> summary(NAME OF MODEL)
```

Look at the output for the model `dipper.phi.sex.p.Timeplussex`. Make sure you understand why there are 5 parameters in this model.

- (j) Look at the output for the model `dipper.phi.sex.p.dot`. Do the males or females have a higher survival rate?
- (k) By looking at the output from the two nested models `dipper.phi.sex.p.dot` and `dipper.phi.dot.p.dot`, conduct a likelihood-ratio test to see if the male and female survivals are significantly different.
- (l) To compare the results from all of the models you have fitted you need to collect them together:

```
> dipper.cjs.results=collect.models()
```

This command looked through all of the objects in the workspace and collected any object that had a class 'mark'.

- (m) Display the list of all models fitted:

```
> dipper.cjs.results
```

- (n) The models in this list are ranked by AIC. Out of the 16 models you have fitted, which is the most appropriate for the data?

Question 2: Analysing simulated data

A data set called `testdata.inp` is located in the Program Mark folder. Copy this data set to an empty directory. This data set is currently in '*.inp' format and so it needs to be converted into a format useable by RMark. Follow the start up instructions for RMark and once you have installed the RMark library read in the data:

```
> testdata=convert.inp("Z:/.../testdata.inp",  
  group.df=data.frame(sex=c("Female","Male")))
```

Following a similar model notation to the dipper and cormorant examples, fit appropriate sex and time-dependent models to the data and using the AIC as a suitable measure, decide from which model the data were simulated.