

Mackerel tagging data

– Procedure to convert from original (Excel) format to model (SAM) format –

Arni Magnusson

15 November 2013

Contents

1	Introduction	1
1.1	Background	1
1.2	Data format specifications	2
2	Conversion procedure	3
2.1	Prepare text file	3
2.2	Process data in R	3
3	R source code	4

1 Introduction

1.1 Background

At the mackerel workshop in Bergen (12–15 Nov 2013), an effort was made to include the available tagging data in a stock assessment model.

The tagging data were supplied by Aril Slotte (IMR, Bergen) who worked with Arni Magnusson (Hafro, Reykjavik) and Dankert Skagen (Bergen) to update the dataset, adding both newer and older entries than were included in the analysis of Tenningen et al. (2011).

Before incorporating them into a stock assessment model, the data needed to be converted from a sparse and highly irregular (non-rectangular) format to a rectangular table. The conversion procedure was not straightforward and is likely to be repeated in the future, after updating again with new releases and recaptures.

The purpose of this document is to describe the conversion procedure, both to document what was done and to make it possible to repeat.

1.2 Data format specifications

Input format: Original data as irregular Excel spreadsheet (only a subset is shown).

The screenshot shows an Excel spreadsheet with columns labeled 'Year of recovery' and 'Nscan' for each year from 2002 to 2013. The data is organized into three sections: Yearclass 2000 (rows 121-137), Yearclass 1999 (rows 138-155), and Yearclass 1998 (rows 156-173). Each section has a 'Year of recovery' column and a 'Nscan' column for each year. The 'Nscan' values are generally higher for earlier years and decrease over time. Recovery rates are shown in the 'Year of recovery' column for later years, with values like 0.717, 0.374, and 0.262.

Output format: Model data as rectangular table (only a subset is shown).

ReleaseY	RecaptureY	Yearclass	Nscan	R	r	Type
2002	2003	2000	3759000	752	3.000	1
2002	2004	2000	4517000	752	2.000	1
2002	2005	2000	3529000	752	4.000	1
2002	2006	2000	4694000	752	0.000	1
2002	2007	2000	2181000	752	1.000	1
2002	2008	2000	3019000	752	0.000	1
2002	2009	2000	442000	752	0.000	1
2002	2010	2000	1209000	752	0.000	1
2003	2004	2000	4517000	552	6.000	1
2003	2005	2000	3529000	552	0.000	1
2003	2006	2000	4694000	552	1.000	1
2003	2007	2000	2181000	552	2.000	1
2003	2008	2000	3019000	552	1.000	1
2003	2009	2000	442000	552	0.000	1
2003	2010	2000	1209000	552	1.000	1
2004	2005	2000	3529000	593	3.000	1
2004	2006	2000	4694000	593	0.000	1
2004	2007	2000	2181000	593	3.000	1
2004	2008	2000	3019000	593	2.000	1
2004	2009	2000	442000	593	0.000	1
2004	2010	2000	1209000	593	0.000	1
2006	2007	2000	2181000	1495	6.000	1
2006	2008	2000	3019000	1495	0.000	1
2006	2009	2000	442000	1495	0.000	1
2006	2010	2000	1209000	1495	0.000	1
2007	2008	2000	3019000	844	1.000	1
2007	2009	2000	442000	844	0.000	1
2007	2010	2000	1209000	844	0.000	1
2008	2009	2000	442000	589	1.000	1
2008	2010	2000	1209000	589	0.000	1
2009	2010	2000	1209000	327	0.000	1
2011	2012	2000	1037000	202	0.374	2

Nscan: number of fish scanned of this cohort in this recapture year,
R: number of fish released, r: number of fish recovered,
Type 1 means steel tag and type 2 means PIT tag.

2 Conversion procedure

2.1 Prepare text file

- Open Excel spreadsheet, copy everything into a text file.
- Delete all initial tabs (at beginning of line), trailing tabs (at end of line), and empty lines.
- Save text file as 'tagdata.txt'.

2.2 Process data in R

Overview of R functions

Four R functions were written to convert the tagging data. All of them have comment headers with detailed usage information. The main workhorses are:

```
readYC  import one year class of data from text file
twist   rearrange one year class of data from sparse to rectangular format
```

The other two functions apply the workhorses to process the entire dataset:

```
readAll  call readYC and twist iteratively and then sort the results
writeAll call readAll, filter the results, multiply, and write to file 'tag.dat'
```

All four R functions contain detailed information in their comment header.

Test individual year classes

Before processing the entire dataset, it is advisable to import and rearrange one year class, to check if everything works correctly:

```
yc2000 <- readYC(2000, file="x:/verk/makrill/data/tagdata.txt")
at2000 <- twist(yc2000)
```

Process all data

After confirming that everything works correctly, process the entire text file:

```
writeAll(yearclasses=1974:2010, infile="x:/verk/makrill/data/tagdata.txt",
         outfile="x:/verk/makrill/data/tag.dat", Nmulti=1000)
```

3 R source code

readYC

```
readYC <- function(yearclass, file="x:/verk/makrill/data/tagdata.txt")
#####
###                                                                    #
### Function: readYC                                                    #
###                                                                    #
### Purpose:  Read one year class of tagging data from original format into R      #
###                                                                    #
### Args:     yearclass is a number, e.g. 2000                                #
###           file is a text file containing entire tagging dataset in original format #
###                                                                    #
### Notes:    The text file does not contain commas, initial/trailing tabs, or empty lines #
###           Procedure to create such a text file: export from Excel, delete commas, delete initial tabs, delete #
###           trailing tabs, and delete empty lines                          #
###                                                                    #
### Returns:  Data frame containing tagging data of one year class in original format #
###                                                                    #
### History:  2013-11-15 Arni Magnusson released                            #
###                                                                    #
#####
{
  ## 1 Locate year class in file
  txt <- readLines(file)
  linum <- match(paste("Yearclass",yearclass), txt) # line number containing "Yearclass" & yearclass number
  linxt <- match(paste("Yearclass",yearclass-1), txt) # line number containing "Yearclass" & previous yearclass number
  if(is.na(linxt))
    linxt <- length(txt) + 1

  ## 2 Read table
  skip <- linum + 2
  nrows <- linxt - linum - 3
  output <- read.table(file, fill=TRUE, sep="\t", header=FALSE, skip=skip, nrows=nrows)
  output <- output[-c(2, seq(4,ncol(output),3))]

  ## 3 Set column names
  skip <- linum
  years <- scan(file, what="", quiet=TRUE, skip=skip, nlines=1)
  years <- as.integer(years[seq(4, length(years), 2)])
  labels <- c("RecaptureY", "Nscan", paste0(c("R","r"), rep(years,each=2)))
  names(output) <- labels

  ## 4 Add year class column
  output <- cbind(Yearclass=as.integer(yearclass), output)

  return(output)
}
```

twist

```
twist <- function(x)
#####
###
### Function: twist
###
### Purpose: Rearrange tagging data from original (Excel) format to model (SAM) format
###
### Args: x is a data frame containing tagging data in original (Excel) format
###
### Notes: If original data look like this,
###
###      Yearclass RecaptureY Nscan R2007 r2007 R2006 r2006 R2005 r2005 R2004 r2004 R2003 r2003 R2002 r2002
###      2000      2002      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      752      NA
###      2000      2003 3759      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      3
###      2000      2004 4517      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      NA      2
###      2000      2005 3529      NA      NA      NA      NA      NA      0      NA      NA      NA      NA      NA      NA      4
###      2000      2006 4694      NA      NA      NA      NA      NA      NA      0      NA      NA      NA      NA      NA      0
###      2000      2007 2181      844      NA      NA      NA      NA      6      NA      0      NA      NA      NA      NA      1
###      2000      2008 3019      NA      1      NA      NA      NA      0      NA      0      NA      NA      NA      NA      0
###
###      then this function should output
###
###      ReleaseY RecaptureY Yearclass Nscan R r Type
###      2002      2003      2000 3759 752 3 1
###      2002      2004      2000 4517 752 2 1
###      2002      2005      2000 3529 752 4 1
###      2002      2006      2000 4694 752 0 1
###      2002      2007      2000 2181 752 1 1
###      2002      2008      2000 3019 752 0 1
###      2003      2004      2000 4517 552 6 1
###      2003      2005      2000 3529 552 0 1
###      2003      2006      2000 4694 552 1 1
###      2003      2007      2000 2181 552 2 1
###      2003      2008      2000 3019 552 1 1
###      2004      2005      2000 3529 593 3 1
###      2004      2006      2000 4694 593 0 1
###      2004      2007      2000 2181 593 3 1
###      2004      2008      2000 3019 593 2 1
###      2005      2006      2000 4694 0 0 1
###      2005      2007      2000 2181 0 0 1
###      2005      2008      2000 3019 0 0 1
###      2006      2007      2000 2181 1495 6 1
###      2006      2008      2000 3019 1495 0 1
###      2007      2008      2000 3019 844 1 1
###
### Returns: Data frame containing tagging data in model (SAM) format
###
### History: 2013-11-15 Arni Magnusson released
###
#####
{
  ## 1 Infer dimensions from the number of release years
  relyrs <- names(x)[seq(4,ncol(x),2)]
  relyrs <- as.integer(substring(relyrs, 2))
  recyrs <- relyrs + 1L
  n <- length(relyrs)
  nrow <- sum(1:n)

  ## 2 Calculate each column for model (SAM) format
  ReleaseY <- unlist(mapply(rep, relyrs, 1:length(relyrs)))
  RecaptureY <- unlist(sapply(recyrs, seq, to=max(recyrs)))
  Yearclass <- rep(x$Yearclass[1], nrow)
  Nscan <- x$Nscan[unlist(sapply((n+1):2, ':', n+1))]
```

```

R <- x[-(n+1), grep("R[0-9]",names(x))]
R <- rev(diag(as.matrix(rev(R))))
R <- rep(R, 1:n)
r <- x[-1, grep("r[0-9]",names(x))]
r <- unlist(mapply(function(x,y) x[y:n], r, n:1))
Type <- ifelse(ReleaseY<2011, 1, 2)

## 3 Construct data frame
output <- data.frame(ReleaseY=ReleaseY, RecaptureY=RecaptureY, Yearclass=Yearclass, Nscan=Nscan, R=R, r=r, Type=Type)
output <- output[order(output[1]),]
row.names(output) <- 1:nrow

return(output)
}

```

readAll

```
readAll <- function(yearclasses=1974:2010, file="x:/verk/makrill/data/tagdata.txt")
#####
###
### Function: ReadAll
###
### Purpose: Read all year classes of tagging data from original (Excel) file and rearrange to model (SAM) format
###
### Args: yearclasses is a vector of year classes, e.g. 1974:2010
### file is a text file containing entire tagging dataset in original format
###
### Requires: readYC, twist
###
### Returns: Data frame containing tagging data in model (SAM) format
###
### History: 2013-11-15 Arni Magnusson released
###
#####
{
  ## 1 Read first year class
  output <- twist(readYC(yearclasses[1], file))

  ## 2 Read subsequent year classes
  for(i in 2:length(yearclasses))
    output <- rbind(output, twist(readYC(yearclasses[i], file)))

  ## 3 Sort data frame
  output <- output[order(output[1],output[2],output[3]),]
  row.names(output) <- 1:nrow(output)

  return(output)
}
```

writeAll

```
writeAll <- function(yearclasses=1974:2010, infile="x:/verk/makrill/data/tagdata.txt",
                    outfile="x:/verk/makrill/data/tag.dat", filter=TRUE, Nmulti=1000)
#####
###
### Function: writeAll
###
### Purpose: Read all year classes of tagging data from text file, rearrange, and write to file
###
### Args:    yearclasses is a vector of year classes, e.g. 1974:2009
###         infile is a text file containing entire tagging dataset in original format
###         outfile is a filename to write to
###         filter is whether to exclude rows with NA values, 0 releases, or 0 scanned
###         Nmulti is a multiplier for Nscan, the number of scanned fish
###
### Requires: readAll
###
### Returns: Writes file to disk
###
### History: 2013-11-15 Arni Magnusson released
###
#####
{
  ## 1 Read data from text file
  dat <- readAll(yearclasses, infile)

  ## 2 Exclude rows with NA values, 0 releases, or 0 scanned
  if(filter)
  {
    dat <- dat[apply(dat, 1, function(x) !any(is.na(x))), ]
    dat <- dat[dat$R>0, ]
    dat <- dat[dat$Nscan>0, ]
    row.names(dat) <- 1:nrow(dat)
  }

  ## 3 Multiply Nscan
  dat$Nscan <- dat$Nscan * Nmulti

  ## 4 Write to file
  write.table(dat, file=outfile, sep="\t", row.names=FALSE, quote=FALSE)

  invisible(dat)
}
```