# **openCR** 2.2 Examples

Murray Efford

2022-01-17

## Contents



## Introduction

This vignette applies the functions in the R package **openCR** to various example datasets. Non-spatial analyses are compared to their equivalents in MARK where appropriate (results suppressed if MARK unavailable).

For clarity, functions and data from other packages are sometimes distinguished using '::' notation, as in `secr::RMarkInput`. This is not strictly necessary for **secr** because it is loaded automatically when **openCR** is loaded, making all exported **secr** functions and data available. Warnings are often generated during likelihood maximization. These typically relate to nonidentifiable parameters, which are common in open-population models. Warnings are generally suppressed here to keep the output readable.

Here the new notation 'PLBx' is used for the Pradel–Link–Barker conditional-likelihood models previously given as 'JSSAxCL' (e.g., 'JSSAsecrfCL' becomes 'PLBsecrf').

```
library(openCR)
RMarkOK <- library(RMark, logical.return=TRUE)        # load RMark for MARK comparisons
markedOK <- library(marked, logical.return = TRUE)    # load marked for comparisons
MarkPath <- 'c:/MARK' # may need to customise this
RMarkOK <- RMarkOK && file.exists(paste0(MarkPath, 'mark64.exe'))
testdir <- 'd:/open populations/tests/'
options(digits = 4, width = 90)   # for more readable output
setNumThreads(7)
```

# Non-spatial analyses

## Patuxent meadow voles

Meadow voles (*Microtus pennsylvanicus*) were trapped by Nichols et al. (1984) on a 10 × 10 grid at Patuxent Wildlife Research Center, Maryland, USA. The data relate to trapping for 5 nights a month over 6 months in 1983. The data were used by Pollock et al. (1990) and Williams, Nichols and Conroy (2002) to demonstrate capture–recapture methods. The data are provided by **openCR** in several forms; see `?microtusCH` for details and other references.

### Data summary

Compare Williams, Nichols and Conroy (2002) Table 17.6 (sex-specific)

```
m.array(microtusFCH)  # females
```

```
##    R  2  3  4  5  6 NRecap
## 1 51 40  4  0  0  0      7
## 2 49    34  3  0  0     12
## 3 52       34  1  0     17
## 4 45          31  0     14
## 5 54             45      9
## 6 72                    72
```

```
m.array(microtusMCH)  # males
```

```
##    R  2  3  4  5  6 NRecap
## 1 53 44  1  0  0  0      8
## 2 69    33  4  0  1     31
## 3 48       32  1  0     15
## 4 56          28  4     24
## 5 45             38      7
## 6 76                    76
```

### CJS

Fit the models in Williams, Nichols and Conroy (2002) Table 17.8. The predictor 't' refers to a factor with one level for each primary session (a synonym of 'session' in **openCR**).

```
args <- list(
    list(microtusFMCH, model = list(phi ~ sex+t, p ~ 1)),
    list(microtusFMCH, model = list(phi ~ t,     p ~ 1)),
    list(microtusFMCH, model = list(phi ~ sex+t, p ~ sex)),
    list(microtusFMCH, model = list(phi ~ sex+t, p ~ sex*t)),
    list(microtusFMCH, model = list(phi ~ t,     p ~ sex)),
    list(microtusFMCH, model = list(phi ~ sex+t, p ~ t)),
    list(microtusFMCH, model = list(phi ~ sex*t, p ~ 1)),
    list(microtusFMCH, model = list(phi ~ sex*t, p ~ sex)),
    list(microtusFMCH, model = list(phi ~ t,     p ~ sex*t)),
    list(microtusFMCH, model = list(phi ~ sex+t, p ~ sex+t)),
    list(microtusFMCH, model = list(phi ~ t,     p ~ sex+t)),
    list(microtusFMCH, model = list(phi ~ sex*t, p ~ sex*t)),
    list(microtusFMCH, model = list(phi ~ t,     p ~ t)),
    list(microtusFMCH, model = list(phi ~ sex*t, p ~ t)),
    list(microtusFMCH, model = list(phi ~ sex*t, p ~ sex+t)),
    list(microtusFMCH, model = list(phi ~ sex,   p ~ sex*t)),
    list(microtusFMCH, model = list(phi ~ 1,     p ~ sex*t)),
```

```
    list(microtusFMCH, model = list(phi ~ sex,   p ~ sex+t)),
    list(microtusFMCH, model = list(phi ~ sex,   p ~ t)),
    list(microtusFMCH, model = list(phi ~ 1,     p ~ sex+t)),
    list(microtusFMCH, model = list(phi ~ 1,     p ~ t)),
    list(microtusFMCH, model = list(phi ~ sex,   p ~ 1)),
    list(microtusFMCH, model = list(phi ~ sex,   p ~ sex)),
    list(microtusFMCH, model = list(phi ~ 1,     p ~ sex)),
    list(microtusFMCH, model = list(phi ~ 1,     p ~ 1))
)
# suppress some unsightly warnings
suppressWarnings(fits <- par.openCR.fit(args, ncores = 1))   # ncores=1 is FASTER than ncores=6
# check completion codes - nlm 3 indicates possible problem in 4th and 16th fit
unname(sapply(fits, function(x) x$fit$code))
```

```
##  [1] 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
```

For comparison we adjust the effective sample size to match that used by the original authors (the default in **openCR** is the number of individuals 308 whereas MARK uses the sum of the release cohort sizes 522)

```
ess <- sum(m.array(microtusFMCH, last.session = FALSE)[,'R'])
AIC(fits, criterion = 'AICc', use.rank = TRUE, n = ess)[,1:7]
```

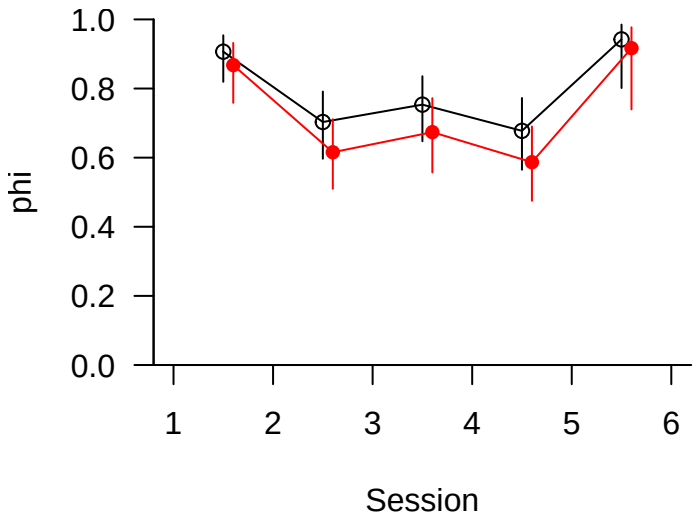
```
##                    model npar rank logLik   AIC  AICc  dAICc
## 1        p~1 phi~sex + t    7    7 -368.6 751.3 751.5  0.000
## 2              p~1 phi~t    6    6 -370.3 752.5 752.7  1.229
## 3      p~sex phi~sex + t    8    8 -368.3 752.5 752.8  1.333
## 5            p~sex phi~t    7    7 -369.6 753.2 753.5  1.979
## 7        p~1 phi~sex * t   11   11 -366.5 754.9 755.4  3.938
## 8      p~sex phi~sex * t   12   12 -365.5 755.0 755.6  4.144
## 4  p~sex * t phi~sex + t   16   14 -362.0 752.1 756.9  5.401
## 6        p~t phi~sex + t   11   10 -367.4 754.9 757.3  5.825
## 9        p~sex * t phi~t   15   12 -363.6 751.3 757.9  6.394
## 10 p~sex + t phi~sex + t   12   11 -366.9 755.7 758.3  6.772
## 11       p~sex + t phi~t   11   10 -368.1 756.2 758.6  7.125
## 13             p~t phi~t   10    9 -369.3 756.5 758.9  7.402
## 14       p~t phi~sex * t   15   14 -365.4 758.8 761.7 10.172
## 15 p~sex + t phi~sex * t   16   15 -364.5 759.0 762.0 10.495
## 12 p~sex * t phi~sex * t   20   17 -360.8 755.5 762.7 11.245
## 16     p~sex * t phi~sex   12    9 -373.7 765.4 771.7 20.225
## 17       p~sex * t phi~1   11    8 -375.1 766.3 772.5 21.053
## 18     p~sex + t phi~sex    8    7 -378.5 770.9 773.2 21.671
## 19           p~t phi~sex    7    6 -379.7 771.5 773.6 22.134
## 20       p~sex + t phi~1    7    6 -380.0 772.0 774.1 22.666
## 21             p~t phi~1    6    5 -381.4 772.8 774.9 23.438
## 22           p~1 phi~sex    3    3 -385.3 776.6 776.6 25.150
## 23         p~sex phi~sex    4    4 -384.5 777.0 777.1 25.617
## 24           p~sex phi~1    3    3 -386.1 778.2 778.2 26.735
## 25             p~1 phi~1    2    2 -387.2 778.4 778.4 26.900
```

Although there is much agreement, AICc values and rankings differ from Williams, Nichols and Conroy (2002) particularly for models such as $(\phi_{s+t}, p_{s*t})$ for which MARK counts parameters differently.

Compare Williams, Nichols and Conroy Fig. 17.2:

```
par(mar = c(4,4,2,2))
maledat <- expand.grid(sex = factor('M', levels = c('F','M')), t = factor(1:6))
```

```
plot(fits[[1]], ylim=c(0,1), type = 'o')
plot(fits[[1]], newdata = maledat, add = TRUE, xoffset = 0.1, col = 'red', pch = 16,
     type = 'o')
```




**CJS by sex**

Compare Williams, Nichols and Conroy Table 17.7:

```
fitfm4 <- openCR.fit(microtusFMCH, model = list(p~t*sex, phi~t*sex))
predict(fitfm4, all = TRUE)
```

```
## $p
##    session sex estimate SE.estimate        lcl    ucl
## 1        1   F       NA          NA         NA     NA
## 2        2   F   0.8831   5.481e-02  7.274e-01 0.9553
## 3        3   F   0.8950   5.708e-02  7.216e-01 0.9656
## 4        4   F   0.9622   3.700e-02  7.759e-01 0.9947
## 5        5   F   1.0000   1.281e-06 1.569e-292 1.0000
## 6        6   F   0.8845          NA         NA     NA
## 7        1   M       NA          NA         NA     NA
## 8        2   M   0.9604   3.872e-02  7.675e-01 0.9944
## 9        3   M   0.8238   7.095e-02  6.420e-01 0.9242
## 10       4   M   0.9114   5.941e-02  7.087e-01 0.9775
## 11       5   M   0.8304   7.482e-02  6.335e-01 0.9328
## 12       6   M   0.9279          NA         NA     NA
##
## $phi
##    session sex estimate SE.estimate    lcl    ucl
## 1        1   F   0.8882     0.05177 0.7409 0.9566
## 2        2   F   0.7819     0.06582 0.6272 0.8843
## 3        3   F   0.6811     0.06635 0.5399 0.7953
## 4        4   F   0.6889     0.06901 0.5409 0.8063
## 5        5   F   0.9421          NA     NA     NA
## 6        6   F       NA          NA     NA     NA
## 7        1   M   0.8645     0.05246 0.7262 0.9388
## 8        2   M   0.5828     0.06554 0.4517 0.7032
## 9        3   M   0.7146     0.07238 0.5554 0.8339
## 10       4   M   0.5869     0.06852 0.4495 0.7120
```

```
## 11       5   M   0.9101          NA     NA     NA
## 12       6   M       NA          NA     NA     NA
```

And in MARK:

```
fm <- secr::RMarkInput(microtusFMCH)
markFM <- RMark::mark(fm, model.parameters =
    list(Phi = list(formula = ~time+sex), p = list(formula = ~time*sex)),
    groups = 'sex', invisible = TRUE, output = FALSE, adjust = FALSE)
round(markFM$results$real[,1:4],4)
```

### Combined-sex meadow vole data from JOLLY

Compare Pollock et al. (1990) Table 4.7

```
JS.counts(microtusCH)
```

```
##     n   R  m  r z
## 1 108 105  0 89 0
## 2 127 121 84 76 5
## 3 102 101 73 68 8
## 4 103 102 73 63 3
## 5 102 100 61 84 5
## 6 149 148 89  0 0
```

[$r_1 = 87$ in the published table, but this is an error]

### Pradel models

Using the formulae of Pradel (1996)[1] and the alternative parameterisations in terms of seniority (gamma) or population growth rate (lambda). The data are for adult male meadow voles at Patuxent.

```
# gamma parameterisation
fitmpradelg <- openCR.fit(microtusMCH, type = "Pradelg",
                          model = list(p~t, phi~t, gamma~t))
# gamma parameterisation, constant gamma
fitmpradelg1 <- openCR.fit(microtusMCH, type = "Pradelg",
                           model = list(p~t, phi~t, gamma~1))
```

Compare Williams et al. Table 18.4. Manually derived $\lambda_i = \phi_i/\gamma_{i+1}$ (SE not calculated).

```
pg <- predict(fitmpradelg)
pg1 <- predict(fitmpradelg1)
pg$phi[5:6,] <- NA
pg1$phi[5:6,] <- NA
df <- cbind(pg$gamma[,2:3], pg1$gamma[,2:3])
names(df) <- c('gamma','segamma','gamma1','se(gamma1)')
df[2,1:2] <- NA
df$lambda <- pg$phi$estimate / c(df$gamma[-1], NA)
df$lambda1 <- pg1$phi$estimate / c(df$gamma1[-1], NA)
df[,c(1,2,5,3,4,6)]
```

```
##    gamma segamma lambda gamma1 se(gamma1) lambda1
## 1     NA      NA     NA     NA         NA  1.3169
## 2     NA      NA 0.8265 0.6524    0.03038  0.8707
## 3 0.7052 0.06760 1.0694 0.6524    0.03038  1.1014
## 4 0.6682 0.06997 0.9001 0.6524    0.03038  0.9284
```

[1] See the help page for rev.capthist (?rev.capthist) for comment on reverse-time formulations.

```
## 5 0.6521 0.07513     NA 0.6524    0.03038      NA
## 6 0.5965 0.06334     NA 0.6524    0.03038      NA
```

The Pradel model estimates the same parameters as the conditional-likelihood JSSA models, but deals differently with losses on capture. Estimates of p amd phi are virtually identical in this example. Estimates of lambda differ because the Pradel lambda estimates include losses on capture. Pradel (1996) suggested a more elaborate formulation to model losses, but this has not been implemented in **openCR**. The JSSA version is preferred if there are losses on capture.

```
# lambda parameterisation
fitmpradel <- openCR.fit(microtusMCH, type = "Pradel",
                         model = list(p~t, phi~t, lambda~t))
fitmPLB1 <- openCR.fit(microtusMCH, type = "PLB1",
                           model = list(p~t, phi~t, lambda~t))
JS <- JS.direct(microtusMCH)
df <- cbind(do.call(rbind, predict(fitmpradel))[,2:3],
            do.call(rbind, predict(fitmPLB1))[,2:3])
names(df) <- c('Pradel','sePradel','JSSA','seJSSA')
df$JS <- c(JS$p, JS$phi, c(JS$N[-1]/JS$N[-6], NA))
df[c(2:5,7:10,14:16),]   # dropping nonidentifiable parameters
```

```
##          Pradel sePradel   JSSA  seJSSA     JS
## p.2      0.9604  0.03872 0.9604 0.03872 0.9608
## p.3      0.8238  0.07095 0.8238 0.07095 0.8251
## p.4      0.9114  0.05942 0.9114 0.05941 0.9124
## p.5      0.8304  0.06878 0.8304 0.06878 0.8310
## phi.1    0.8644  0.05247 0.8644 0.05246 0.8641
## phi.2    0.5828  0.06554 0.5828 0.06554 0.5820
## phi.3    0.7146  0.07238 0.7146 0.07238 0.7147
## phi.4    0.5869  0.06852 0.5869 0.06852 0.5870
## lambda.2 0.8264  0.10462 0.8089 0.10027 0.7924
## lambda.3 1.0695  0.13865 1.0453 0.13193 1.0507
## lambda.4 0.9001  0.12408 0.8808 0.11883 0.8840
```

One would expect the difference to disappear if there are no losses. We construct an artificial dataset that drops these capture histories entirely, and compare estimates across models.

```
tmp <- microtusMCH
tmp <- tmp[apply(tmp,1,min)>=0,,, drop = FALSE]
class(tmp) <- 'capthist'
pradelnoloss <- openCR.fit(tmp, type = "Pradel",
                           model = list(p~t, phi~t, lambda~t))
JSSAnoloss   <- openCR.fit(tmp, type = "PLB1",
                           model = list(p~t, phi~t, lambda~t))

df <- cbind(do.call(rbind, predict(pradelnoloss))[,2:3],
            do.call(rbind, predict(JSSAnoloss))[,2:3])
names(df) <- c('Pradel','sePradel','JSSA','seJSSA')
```

```
# also run Pradel model in MARK
captdf <- RMarkInput(tmp, grouped = TRUE, covariates = FALSE)
time <- list(formula = ~time)
mk <- mark(captdf, model = "Pradlambda", model.parameters =
              list(Phi = time, p = time, Lambda = time), invisible = TRUE)
dfmk <- mk$results$real[c(7:10, 1:4, 13:15),1:2]
names(dfmk) <- c('MKPradel','seMKPradel')
cbind(df[c(2:5,7:10,14:16),], dfmk)   # dropping nonidentifiable parameters
```

### JSSA conditional likelihood with per capita recruitment

This model (JSSAfCL aka PLBf) coincides with the Link and Barker (2005) formulation, as provided in MARK. We compare the two.

```
fitf1 <- openCR.fit(microtusMCH, type= "PLBf", model = list(p~1, phi~1, f~1))
summary(fitf1)
```

```
## $versiontime
## [1] "2.2.2, run 12:31:31 17 Jan 2022"
##
## $capthist
##             S1 S2 S3 S4 S5 S6 Total
## Occasions    1  1  1  1  1  1     6
## Detections 56 72 49 57 46 77   357
## Animals    56 72 49 57 46 77   171
##
## $modeldetails
##  type fixed distribution
##  PLBf  none         none
##
## $AICtable
##          model npar rank logLik  AIC AICc
##  p~1 phi~1 f~1    3    3 -507.4 1021 1022
##
## $link
##      p   phi   f
##  logit logit log
##
## $coef
##        beta SE.beta     lcl     ucl
## p    2.1434  0.2878  1.5793  2.7075
## phi  0.9351  0.1330  0.6744  1.1958
## f   -1.0797  0.1008 -1.2772 -0.8822
##
## $hessian
##  rankH svtol Eigen1 Eigen2  Eigen3
##      3 1e-05      1 0.3432 0.08176
##
## $predicted
## $predicted$p
##   session estimate SE.estimate    lcl    ucl
## 1       1    0.895     0.02704 0.8291 0.9375
## 2       2    0.895     0.02704 0.8291 0.9375
## 3       3    0.895     0.02704 0.8291 0.9375
## 4       4    0.895     0.02704 0.8291 0.9375
## 5       5    0.895     0.02704 0.8291 0.9375
## 6       6    0.895     0.02704 0.8291 0.9375
##
## $predicted$phi
##   session estimate SE.estimate    lcl    ucl
## 1       1   0.7181     0.02693 0.6625 0.7678
## 2       2   0.7181     0.02693 0.6625 0.7678
```

```
## 3       3   0.7181     0.02693 0.6625 0.7678
## 4       4   0.7181     0.02693 0.6625 0.7678
## 5       5   0.7181     0.02693 0.6625 0.7678
## 6       6       NA          NA     NA     NA
##
## $predicted$f
##   session estimate SE.estimate    lcl    ucl
## 1       1   0.3397     0.03423 0.2788 0.4139
## 2       2   0.3397     0.03423 0.2788 0.4139
## 3       3   0.3397     0.03423 0.2788 0.4139
## 4       4   0.3397     0.03423 0.2788 0.4139
## 5       5   0.3397     0.03423 0.2788 0.4139
## 6       6       NA          NA     NA     NA
```

And in MARK:

```
df <- RMarkInput(microtusMCH, grouped = TRUE, covariates = FALSE)
mark(df, model = "LinkBarker")
```

MARK generates files that we don't need, so we call on **RMark** to clean up.

```
RMark::cleanup(ask = FALSE)
```

### Direct (closed-form) Jolly-Seber estimates

The Jolly-Seber method in its original form uses sufficient statistics to quickly compute population size, apparent survival and recruitment. Here we use the combined-sex dataset derived from JOLLY. Compare Pollock et al. (1990) Table 4.8:

```
JS.direct(microtusCH)
```

```
##     n   R  m  r z      p     sep     N   seN    phi   sephi    covphi     B   seB
## 1 108 105  0 89 0     NA      NA    NA    NA 0.8754 0.03862 -0.001669    NA    NA
## 2 127 121 84 76 5 0.9138 0.03667 138.4 4.145 0.6580 0.04756 -0.001159 30.94 3.563
## 3 102 101 73 68 8 0.8606 0.04545 118.1 4.409 0.6898 0.04881 -0.001226 28.63 2.836
## 4 103 102 73 63 3 0.9380 0.03458 109.4 2.928 0.6266 0.04902        NA 43.30 3.017
## 5 102 100 61 84 5 0.9112 0.03779 111.2 3.126     NA      NA        NA    NA    NA
## 6 149 148 89  0 0     NA      NA    NA    NA     NA      NA        NA    NA    NA
```

## Dippers

Lebreton et al. (1992) demonstrated Cormack-Jolly-Seber methods with a dataset on European Dipper (*Cinclus cinclus*) collected by Marzolin (1988) and the data have been much used since then. Dippers were captured annually over 1981–1987.

### Data summary

Compare Lebreton et al. 1992 Table 10:

```
dipperCHM <-  subset(dipperCH, covariates(dipperCH)$sex=='Male')
dipperCHF <-  subset(dipperCH, covariates(dipperCH)$sex=='Female')
m.array(dipperCHM)
```

```
##       R 1982 1983 1984 1985 1986 1987 NRecap
## 1981 12    6    1    0    0    0    0      5
## 1982 26        11    0    0    0    0     15
## 1983 37             17    1    0    0     19
## 1984 39                  22    0    1     16
```

```
## 1985 45                       25    0     20
## 1986 48                            28     20
## 1987 46                                   46
```

```
m.array(dipperCHF)
```

```
##       R 1982 1983 1984 1985 1986 1987 NRecap
## 1981 10    5    1    0    0    0    0      4
## 1982 34        13    1    0    0    0     20
## 1983 41             17    1    0    0     23
## 1984 41                  23    1    1     16
## 1985 43                       26    0     17
## 1986 50                            24     26
## 1987 47                                   47
```

Test 3.SR (Lebreton et al. 1992 p. 86). Lebreton et al. indicate that only component 3SR is meaningful.

```
test3sr <- rbind(
    Male = ucare.cjs(dipperCHM)$test3sr$test3sr,
    Female = ucare.cjs(dipperCHF)$test3sr$test3sr
)[,1:3]
test3sr <- rbind(test3sr, Total = apply(test3sr,2,sum))
test3sr[,3] <- 1 - pchisq(test3sr[,1], test3sr[,2])
test3sr
```

```
##          stat df  p_val
## Male    6.778  5 0.2377
## Female  4.985  5 0.4177
## Total  11.763 10 0.3012
```

### CJS models

Compare Lebreton et al. (1992) Table 11

```
dipper.p.t.phi.t <- openCR.fit(dipperCH, model = list(p~t, phi~t))
predict(dipper.p.t.phi.t)
```

```
## $p
##   session estimate SE.estimate    lcl    ucl
## 1    1981       NA          NA     NA     NA
## 2    1982   0.6962     0.16578 0.3303 0.9142
## 3    1983   0.9231     0.07288 0.6161 0.9890
## 4    1984   0.9130     0.05818 0.7140 0.9779
## 5    1985   0.9008     0.05384 0.7360 0.9673
## 6    1986   0.9324     0.04581 0.7685 0.9829
## 7    1987   0.7432          NA     NA     NA
##
## $phi
##   session estimate SE.estimate    lcl    ucl
## 1    1981   0.7182     0.15556 0.3610 0.9200
## 2    1982   0.4347     0.06883 0.3075 0.5711
## 3    1983   0.4782     0.05971 0.3644 0.5943
## 4    1984   0.6261     0.05927 0.5048 0.7334
## 5    1985   0.5985     0.05605 0.4855 0.7019
## 6    1986   0.7140          NA     NA     NA
## 7    1987       NA          NA     NA     NA
```

The individual covariate 'sex' is a factor with levels 'Female' and 'Male'. We can fit Cormack-Jolly-Seber models directly with `openCR.fit`:

```
dipper.null <- openCR.fit(dipperCH)
dipper.p.sex <- openCR.fit(dipperCH, model = p~sex)
dipper.phi.t <- openCR.fit(dipperCH, model = phi~t)
dipper.phi.sex.p.t.sex <- openCR.fit(dipperCH, model = list(phi~sex, p~t+sex))
```

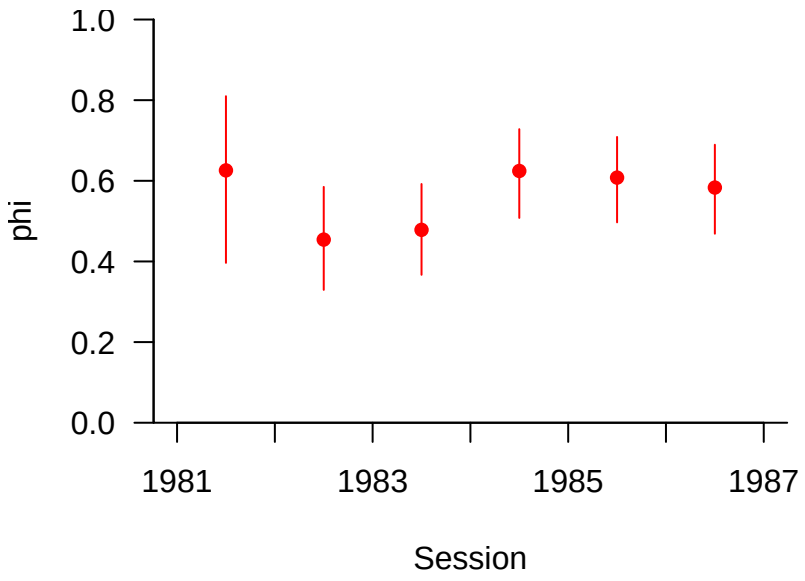
```
AIC(dipper.null, dipper.p.sex, dipper.phi.sex.p.t.sex)
```

```
##                                    model npar rank logLik   AIC  AICc   dAIC  AICwt
## dipper.null                      p~1 phi~1    2    2 -333.4 670.8 671.1  0.000 0.6632
## dipper.p.sex                   p~sex phi~1    3    3 -333.1 672.2 672.7  1.355 0.3368
## dipper.phi.sex.p.t.sex p~t + sex phi~sex    9    9 -331.9 681.9 685.9 11.016 0.0000
```

```
predict(dipper.p.sex, all.levels = TRUE)
```

```
## $p
##    session    sex estimate SE.estimate    lcl    ucl
## 1     1981   Male       NA          NA     NA     NA
## 2     1982   Male   0.9243     0.03542 0.8191 0.9705
## 3     1983   Male   0.9243     0.03542 0.8191 0.9705
## 4     1984   Male   0.9243     0.03542 0.8191 0.9705
## 5     1985   Male   0.9243     0.03542 0.8191 0.9705
## 6     1986   Male   0.9243     0.03542 0.8191 0.9705
## 7     1987   Male   0.9243     0.03542 0.8191 0.9705
## 8     1981 Female       NA          NA     NA     NA
## 9     1982 Female   0.8801     0.04345 0.7661 0.9427
## 10    1983 Female   0.8801     0.04345 0.7661 0.9427
## 11    1984 Female   0.8801     0.04345 0.7661 0.9427
## 12    1985 Female   0.8801     0.04345 0.7661 0.9427
## 13    1986 Female   0.8801     0.04345 0.7661 0.9427
## 14    1987 Female   0.8801     0.04345 0.7661 0.9427
##
## $phi
##    session    sex estimate SE.estimate    lcl    ucl
## 1     1981   Male   0.5607     0.02516 0.5109 0.6093
## 2     1982   Male   0.5607     0.02516 0.5109 0.6093
## 3     1983   Male   0.5607     0.02516 0.5109 0.6093
## 4     1984   Male   0.5607     0.02516 0.5109 0.6093
## 5     1985   Male   0.5607     0.02516 0.5109 0.6093
## 6     1986   Male   0.5607     0.02516 0.5109 0.6093
## 7     1987   Male       NA          NA     NA     NA
## 8     1981 Female   0.5607     0.02516 0.5109 0.6093
## 9     1982 Female   0.5607     0.02516 0.5109 0.6093
## 10    1983 Female   0.5607     0.02516 0.5109 0.6093
## 11    1984 Female   0.5607     0.02516 0.5109 0.6093
## 12    1985 Female   0.5607     0.02516 0.5109 0.6093
## 13    1986 Female   0.5607     0.02516 0.5109 0.6093
## 14    1987 Female       NA          NA     NA     NA
```

```
par(mar = c(4,4,2,2))
plot(dipper.phi.t, par = 'phi', ylim = c(0,1), pch = 16, col = 'red')
```




Comparing to Laake, Johnson and Conn (2013) analysis of dummy 'dipper' data in MEE

```
# go to 'marked' to get exact copy of dipper data
# and add a dummy weight field which are random values from 1 to 10
data(dipper)    # assumes marked already loaded
set.seed(123)
dipper$weight <- round(runif(nrow(dipper),0,9),0)+1
# switch to 'openCR' and fit model
dipperCH2 <- unRMarkInput(dipper)  # convert to secr capthist object
sesscov <- data.frame(flood = c(0,1,1,0,0,0,0))   ## extra 0 to complete 7-session vector
fit2 <- openCR.fit(dipperCH2, model = list(p ~ sex + B, phi ~ flood + weight),
                   sessioncov = sesscov)
coef(fit2)[c(4,5,6,1,2,3),]
```

```
##                  beta SE.beta      lcl      ucl
## phi          0.524925 0.28203 -0.02784  1.07769
## phi.flood   -0.581408 0.22850 -1.02926 -0.13355
## phi.weight  -0.003354 0.04029 -0.08231  0.07561
## p           -0.156994 1.18256 -2.47478  2.16079
## p.sexMale    0.460348 0.72270 -0.95612  1.87682
## p.BTRUE      1.966771 0.88543  0.23136  3.70218
```

This does not exactly reproduce Laake, Johnson and Conn (2013) Table 5 because they discount first captures as the basis for trap dependence (without explanation). If their analysis is modified by dropping the line `td[releaseocc]=0` then results agree exactly.

### JSSA models

```
dipper.JSSA <- openCR.fit(dipperCH, type='JSSAl', model = list(phi~t, lambda~t))
predict(dipper.JSSA)$lambda
```

```
##   session estimate SE.estimate    lcl   ucl
## 1    1981    2.792     0.58050 1.8573 4.196
## 2    1982    1.265     0.17102 0.9709 1.649
## 3    1983    1.026     0.12172 0.8132 1.295
## 4    1984    1.104     0.11275 0.9038 1.349
## 5    1985    1.103     0.10876 0.9089 1.338
```

```
## 6    1986    0.958     0.09356 0.7911 1.160
## 7    1987       NA          NA     NA    NA
```

### Compare JS in openCR and marked

For exact comparison the distribution of $n$ must be changed from Poisson (the default in `openCR.fit`) to binomial (the assumed distribution in MARK and **marked**).

```
## openCR
coef(openCR.fit(dipperCH, type = 'JSSAb', distribution = 'binomial'))
```

```
## Warning in log(1 - prob): NaNs produced
```

```
##          beta SE.beta    lcl    ucl
## p      2.2914  0.3324 1.6399 2.9430
## phi    0.2383  0.1016 0.0391 0.4374
## b      0.6683  0.2233 0.2306 1.1061
## superN 2.7196  0.4295 1.8778 3.5613
```

```
## marked
data(dipper)
crm(dipper, model="js")
```

```
## Starting optimization 4  parameters
##  Number of evaluations:  100  -2lnl: -958.5951007 Number of evaluations:  200  -2lnl: -958.7091896
```

```
##
## crm Model Summary
##
## Npar :  4
## -2lnL:  705.6
## AIC  :  713.6
##
## Beta
##                   Estimate
## Phi.(Intercept)     0.2383
## p.(Intercept)       2.2915
## pent.(Intercept)    0.6683
## N.(Intercept)       2.7196
```

Coefficients are numerically identical. Likelihood and AIC values differ between **openCR** and **marked** because **openCR** omits a constant term.

### Compare estimates from different models

```
cjs     <- openCR.fit(dipperCH, type = 'CJS',     model = list(p~t, phi~t))
plbb <- openCR.fit(dipperCH, type = 'PLBb', model = list(p~t, phi~t, b~t))
plbf <- openCR.fit(dipperCH, type = 'PLBf', model = list(p~t, phi~t, f~t))
plbg <- openCR.fit(dipperCH, type = 'PLBg', model = list(p~t, phi~t, gamma~t))
plbl <- openCR.fit(dipperCH, type = 'PLBl', model = list(p~t, phi~t, lambda~t),
                   start = plbg)
jssab   <- openCR.fit(dipperCH, type = 'JSSAb',   model = list(p~t, phi~t, b~t))
jssaf   <- openCR.fit(dipperCH, type = 'JSSAf',   model = list(p~t, phi~t, f~t))
jssag   <- openCR.fit(dipperCH, type = 'JSSAg',   model = list(p~t, phi~t, gamma~t))
jssal   <- openCR.fit(dipperCH, type = 'JSSAl',   model = list(p~t, phi~t, lambda~t))
jssaN   <- openCR.fit(dipperCH, type = 'JSSAN',   model = list(p~t, phi~t, N~t))
# combine as openCRlist
```

```
fits <- openCRlist(cjs, plbb, plbf, plbg, plbl, jssab, jssaf, jssag,
                   jssal, jssaN)
```

```
make.table(fits[-1], parm = "p")
```

```
##        session
## model     1981   1982   1983   1984   1985   1986   1987
##   plbb  0.8473 0.6962 0.9231 0.9131 0.9008 0.9324 0.8146
##   plbf  0.7596 0.6962 0.9231 0.9130 0.9008 0.9324 0.7231
##   plbg  0.7772 0.6962 0.9231 0.9130 0.9008 0.9324 0.7777
##   plbl  0.7338 0.6962 0.9231 0.9130 0.9008 0.9324 0.7273
##   jssab 0.8483 0.6962 0.9231 0.9130 0.9008 0.9324 0.8256
##   jssaf 0.7605 0.6962 0.9231 0.9130 0.9008 0.9324 0.7163
##   jssag 0.7769 0.6962 0.9231 0.9130 0.9008 0.9324 0.7712
##   jssal 0.8257 0.6962 0.9231 0.9130 0.9008 0.9324 0.8028
##   jssaN 0.8011 0.6962 0.9231 0.9130 0.9008 0.9324 0.8705
```

```
make.table(fits, parm = "phi")
```

```
##        session
## model     1981   1982   1983   1984   1985   1986 1987
##   cjs   0.7182 0.4347 0.4782 0.6261 0.5985 0.7140
##   plbb  0.7182 0.4347 0.4782 0.6261 0.5985 0.6514
##   plbf  0.7182 0.4347 0.4782 0.6261 0.5985 0.7338
##   plbg  0.7182 0.4347 0.4782 0.6261 0.5985 0.6823
##   plbl  0.7181 0.4347 0.4782 0.6261 0.5985 0.7295
##   jssab 0.7182 0.4347 0.4782 0.6261 0.5985 0.6427
##   jssaf 0.7182 0.4347 0.4782 0.6261 0.5985 0.7407
##   jssag 0.7182 0.4347 0.4782 0.6261 0.5985 0.6881
##   jssal 0.7182 0.4347 0.4782 0.6261 0.5985 0.6610
##   jssaN 0.7182 0.4347 0.4782 0.6261 0.5985 0.6095
```

Estimates from the models parameterized with lambda (population growth rate) stand out as slightly different. It is not clear whether this is inherent in the parameterization (lambda is the arithmetic sum of the survival and recruitment components) or to an undetected bug in the code. I'm betting on the former.

Update 2021-08-09: The precedingissue was resolved by increasing the default iterations for nlm(). The lambda parameterizations were just slower to fit.

### Compare Schofield and Barker (2016)

Schofield and Barker (2016 supplementary materials) presented an analysis of the dipper data that compared CJS and re-parameterised Jolly-Seber or POPAN models (CMSA models in their terminology). Their favoured parameterisation uses $\kappa$ (kappa) for recruitment. $\kappa$ has a complex recursive relationship to $p$, $\phi$ and $f$ (their $\gamma$). The $\kappa$ parameterisation enables a revealing factorisation of the likelihood (Link and Barker 2005, Schofield and Barker 2016). It is not so obvious that it has other benefits, and it has not appeared in the applied capture–recapture literature. The kappa parameterisation therefore remains an undocumented feature of **openCR**; we demonstrate it here as a curiosity. For the kappa parameterisation use "k" in the model type (e.g., JSSAk instead of JSSAf) and use the real parameter name "kappa" in model formulae.

We suppress warnings from `openCR.fit` regarding likely confounding - we expect this for $\phi_6$ and $p_7$.

```
cjs <- openCR.fit(dipperCH, type = 'CJS', model = list(p~t, phi~t))
CMSAn <- openCR.fit(dipperCH, type = 'PLBk', model = list(p~t, phi~t, kappa~t))
CMSA <- openCR.fit(dipperCH, type = 'JSSAk', model = list(p~t, phi~t, kappa~t),
                   distribution = "binomial")
CMSApois <- openCR.fit(dipperCH, type = 'JSSAk', model = list(p~t, phi~t, kappa~t),
                   distribution = "poisson")
fits <- openCRlist(cjs, CMSAn, CMSA, CMSApois)
AIC(fits)[,1:4] # logLik directly comparable only between CMSA and CMSApois
```

```
##                               model npar rank logLik
## cjs                        p~t phi~t   12   12 -328.5
## CMSAn              p~t phi~t kappa~t   19   17 -892.7
## CMSA     p~t phi~t kappa~t superN~1   20   18 -895.1
## CMSApois p~t phi~t kappa~t superN~1   20   18 -896.5
```

```
# capture probability
make.table(fits, 'p')
```

```
##           session
## model        1981   1982   1983   1984   1985   1986   1987
##   cjs             0.6962 0.9231 0.9130 0.9008 0.9324 0.7432
##   CMSAn    0.8595 0.6962 0.9231 0.9130 0.9008 0.9324 0.8553
##   CMSA     1.0000 0.7341 0.9270 0.9149 0.9024 0.9338 1.0000
##   CMSApois 0.8595 0.6962 0.9231 0.9130 0.9008 0.9324 0.8552
```

```
# survival
make.table(fits, 'phi')
```

```
##           session
## model        1981   1982   1983   1984   1985   1986 1987
##   cjs      0.7182 0.4347 0.4782 0.6261 0.5985 0.7140
##   CMSAn    0.7182 0.4347 0.4782 0.6261 0.5985 0.6204
##   CMSA     0.6983 0.4387 0.4785 0.6262 0.5987 0.5310
##   CMSApois 0.7182 0.4347 0.4782 0.6261 0.5985 0.6204
```

```
# kappa (recruitment+)
make.table(fits, 'kappa')
```

```
##           session
## model      1981  1982  1983  1984  1985  1986  1987
##   cjs
##   CMSAn         2.227 2.364 2.045 1.864 2.091 1.773
##   CMSA          2.215 2.361 2.044 1.862 2.089 1.773
##   CMSApois      2.227 2.364 2.045 1.864 2.091 1.773
```

```
# superpopulation size
make.table(fits, 'superN')[,1]
```

```
##      cjs    CMSAn     CMSA CMSApois
##       NA       NA    310.4    320.6
```

These estimates (particularly $\kappa_7$) differ from those of Schofield and Barker (2016) in part because of a small error in their copy of the dipper data (10 extra individuals in the last release cohort) (M. Schofield pers. comm.).

A derived (H-T) estimate of superpopulation size may be got from the conditional likelihood model:

```
derived(CMSAn)$superN
```

```
## [1] 320.6
```

This is close to the estimate from the full Poisson model, but not identical. More worrying, the superpopulation estimates change when we remaximise the models with different starting values for p and phi (for example, those from the CJS model):

```
CMSA2 <- openCR.fit(dipperCH, type = 'JSSAk', model = list(p~t, phi~t, kappa~t),
                    distribution = "binomial", start = cjs)
CMSApois2 <- openCR.fit(dipperCH, type = 'JSSAk', model = list(p~t, phi~t, kappa~t),
                    distribution = "poisson", start = cjs)
fits2 <- openCRlist(CMSA2, CMSApois2)
make.table(fits2, 'superN')[,1]
```

```
##     CMSA2 CMSApois2
##     310.4     331.1
```

It's fairly clear that, despite the reparameterisation, the full model with time variation in all parameters remains unidentifiable. Constraints are needed, as we should have expected!

## *Gonodontis* moths

Male moths (the nonmelanic form of *Gonodontis bidentata*) were trapped, marked and released by Bishop et al. (1978). The data were analysed by Crosbie (1979) and Link and Barker (2005, 2010).

### Data summary

```
m.array(gonodontisCH)
```

```
##     R  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 NRecap
## 1  15  8  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0      5
## 2  52    16  5  1  0  0  0  0  0  0  0  0  0  0  0  0  0     30
## 3  54       12  2  0  0  0  0  0  0  0  0  0  0  0  0  0     40
## 4  62          12  2  0  0  1  0  0  0  0  0  0  0  0  0     47
## 5  29             10  1  1  0  1  0  0  0  0  0  0  0  0     16
## 6  84                 5  3  2  1  0  0  0  0  1  0  0  0     72
## 7  51                   15  0  1  0  0  0  0  0  0  0  0     35
## 8  74                       8  4  0  1  1  3  0  0  0  0     57
## 9  43                          7  1  3  0  1  0  0  0  0     31
## 10 85                             2  6  2  2  0  0  0  0     73
## 11 15                                2  1  0  0  0  0  0     12
## 12 81                                  20  3  2  0  0  0     56
## 13 93                                      8  6  3  0  0     76
## 14 59                                         11  1  2  0     45
```

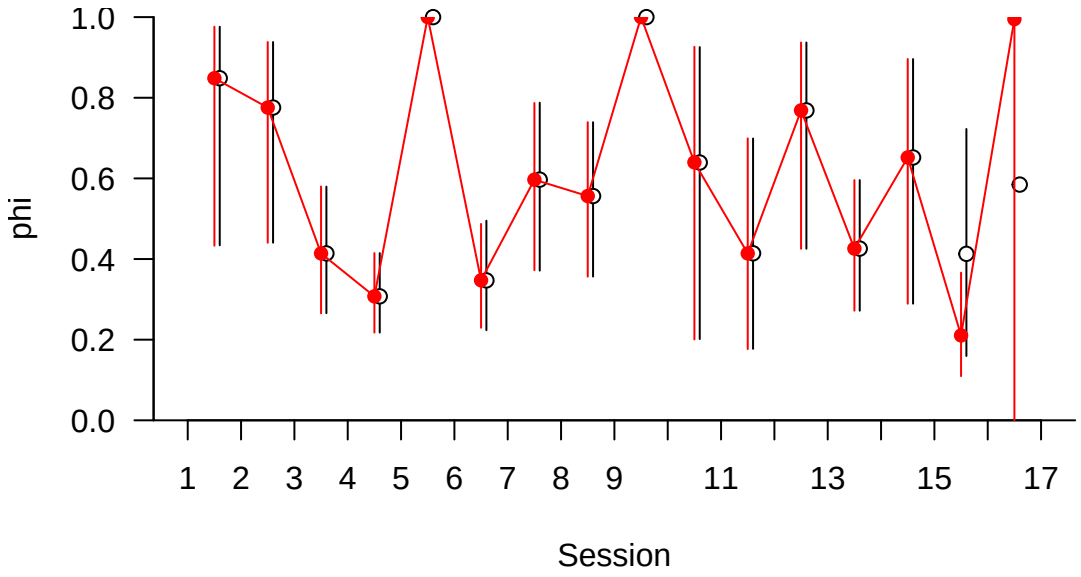
```
## 15 60                                          5  2     53
## 16 37                                             4     33
## 17  8                                                   8
```

**CJS vs JSSA**

```
fitCJS <- openCR.fit(gonodontisCH, model = list(phi~t, p~t))
fitPLBf<- openCR.fit(gonodontisCH, type = 'PLBf',
                     model = list(f~t, phi~t, p~t))
fitJSSAB<- openCR.fit(gonodontisCH, type = 'JSSAB',
                     model = list(BN~t, phi~t, p~t))
```

We choose to plot 80% confidence intervals because the estimates are quite rough.

```
par(mar=c(4,4,2,2))
plot(fitCJS,  yaxs = 'i', ylim = 0:1, alpha = 0.2, xoffset = 0.1,
     pch = 21, bg='white', xpd = TRUE)
plot(fitPLBf, add = TRUE, type='o', col = 'red', alpha = 0.2, pch = 16)
```



## Orongorongo Valley brushtail possums

### Data summary

```
summary(FebpossumCH, terse = TRUE)
```

```
##            1980 1981 1982 1983 1984 1985 1986 1987 1988
## Occasions    10    8    6    4    4    4    5    5    6
## Detections  461  462  423  248  278  214  294  318  275
## Animals     142  146  147  138  129  106  133  141  114
## Detectors     1    1    1    1    1    1    1    1    1
```

```
m.array(FebpossumCH)
```

```
##        R 1981 1982 1983 1984 1985 1986 1987 1988 NRecap
## 1980 142   85    0    0    1    0    0    0    0     56
## 1981 146        95    3    2    0    0    0    0     46
## 1982 147            100    9    3    1    0    0     34
## 1983 138                 98    5    1    0    0     34
## 1984 129                      83    5    3    1     37
## 1985 106                           69    7    2     28
## 1986 133                                85    4     44
## 1987 141                                     86     55
## 1988 114                                           114
```

### CJS analyses

Pledger et al. (2003) analysed data from February samples in 1980–1988, using all captures.

```
# collapse because PPN did not use robust design
FebCH <- reduce(FebpossumCH, by = 'all', verify = FALSE)
m.array(FebCH)  # identical to above
```

```
##        R 1981 1982 1983 1984 1985 1986 1987 1988 NRecap
## 1980 142   85    0    0    1    0    0    0    0     56
## 1981 146        95    3    2    0    0    0    0     46
## 1982 147            100    9    3    1    0    0     34
## 1983 138                 98    5    1    0    0     34
## 1984 129                      83    5    3    1     37
## 1985 106                           69    7    2     28
## 1986 133                                85    4     44
## 1987 141                                     86     55
## 1988 114                                           114
```

```
FebCHF <- subset(FebCH, function(x) covariates(x)$sex=='F')
FebCHM <- subset(FebCH, function(x) covariates(x)$sex=='M')

baseargs <- list(capthist = 'FebCHF', type = 'CJS')
args <- rep(list(baseargs),24)
args[[1]]$model <- list(p~t, phi~t)
args[[2]]$model <- list(p~t+h2, phi~t)
args[[3]]$model <- list(p~t*h2, phi~t)
args[[4]]$model <- list(p~t, phi~t+h2)
args[[5]]$model <- list(p~t+h2, phi~t+h2)
args[[6]]$model <- list(p~t, phi~t*h2)
args[[7]]$model <- list(p~t+h2, phi~t*h2)
args[[8]]$model <- list(p~t*h2, phi~t*h2)
```

```
for (i in 9:16) {
    args[[i]] <- args[[i-8]]
    args[[i]]$capthist <- 'FebCHM'
}
for (i in 17:24) {
    args[[i]] <- args[[i-8]]
    args[[i]]$capthist <- 'FebCH'
}

fitsCJS <- par.openCR.fit(args, ncores = 7)
```

```
## Warning in par.openCR.fit(args, ncores = 7): par.secr.fit with ncores > 1 is SLOWER than
## par.secr.fit with ncores = 1
```

```
## Completed in 2.134 minutes at 12:35:43 17 Jan 2022
```

```
# Females
AIC(openCRlist(fitsCJS[1:8]), sort = FALSE, criterion = 'AIC')[,-(5:6)]
```

```
##                         model npar rank logLik   dAIC  AICwt
## 1                   p~t phi~t   16   14 -366.9  3.979 0.0529
## 2      p~t + h2 phi~t pmix~h2   18   16 -362.9  0.000 0.3867
## 3      p~t * h2 phi~t pmix~h2   25   19 -359.2  6.731 0.0134
## 4      p~t phi~t + h2 pmix~h2   18   15 -363.3  0.814 0.2574
## 5 p~t + h2 phi~t + h2 pmix~h2   19   17 -362.3  0.825 0.2560
## 6      p~t phi~t * h2 pmix~h2   25   16 -362.4 13.135 0.0000
## 7 p~t + h2 phi~t * h2 pmix~h2   26   18 -357.3  4.889 0.0336
## 8 p~t * h2 phi~t * h2 pmix~h2   33   20 -354.6 13.397 0.0000
```

```
# Males
AIC(openCRlist(fitsCJS[9:16]), sort = FALSE, criterion = 'AIC')[,-(5:6)]
```

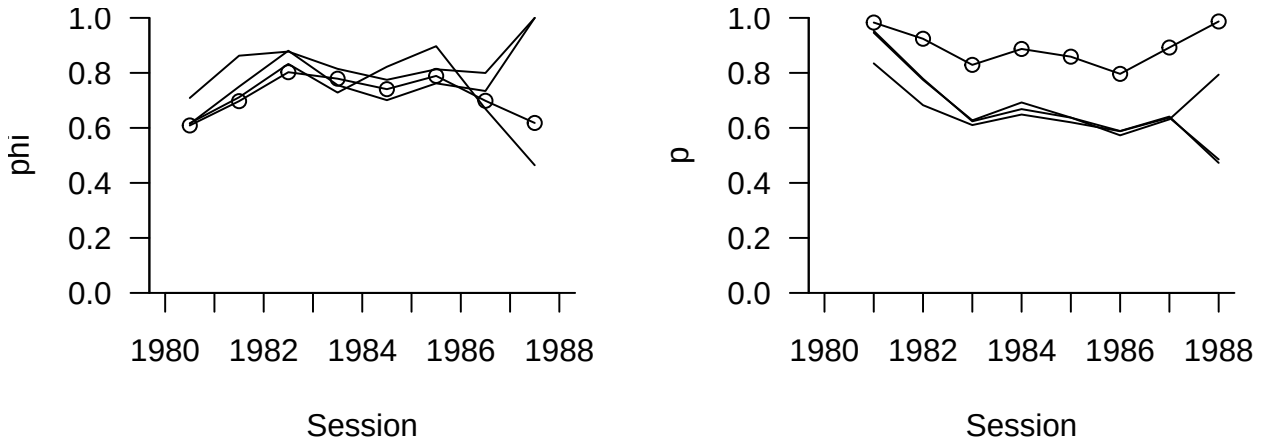
```
##                         model npar rank logLik   dAIC  AICwt
## 1                   p~t phi~t   16   15 -483.3 31.092 0.0000
## 2      p~t + h2 phi~t pmix~h2   18   17 -468.5  5.486 0.0454
## 3      p~t * h2 phi~t pmix~h2   25   17 -468.5 19.486 0.0000
## 4      p~t phi~t + h2 pmix~h2   18   16 -470.3  9.147 0.0073
## 5 p~t + h2 phi~t + h2 pmix~h2   19   17 -464.7  0.000 0.7057
## 6      p~t phi~t * h2 pmix~h2   25   18 -466.1 14.757 0.0000
## 7 p~t + h2 phi~t * h2 pmix~h2   26   22 -458.8  2.144 0.2416
## 8 p~t * h2 phi~t * h2 pmix~h2   33   22 -458.8 16.144 0.0000
```

```
# Combined
AIC(openCRlist(fitsCJS[17:24]), sort = FALSE, criterion = 'AIC')[,-(5:6)]
```

```
##                         model npar rank logLik   dAIC  AICwt
## 1                   p~t phi~t   16   15 -867.0 46.587 0.0000
## 2      p~t + h2 phi~t pmix~h2   18   17 -842.3  1.293 0.3403
## 3      p~t * h2 phi~t pmix~h2   25   19 -839.9 10.443 0.0000
## 4      p~t phi~t + h2 pmix~h2   18   16 -849.3 15.137 0.0000
## 5 p~t + h2 phi~t + h2 pmix~h2   19   18 -840.7  0.000 0.6496
## 6      p~t phi~t * h2 pmix~h2   25   17 -848.3 27.221 0.0000
## 7 p~t + h2 phi~t * h2 pmix~h2   26   23 -837.9  8.332 0.0101
## 8 p~t * h2 phi~t * h2 pmix~h2   33   26 -837.1 20.811 0.0000
```

```
pred18 <- predict(fitsCJS[[18]], all.levels = TRUE)
pred21 <- predict(fitsCJS[[21]], all.levels = TRUE)
```

```
pred23 <- predict(fitsCJS[[23]], all.levels = TRUE)
avp18 <- pred18$pmix$estimate[1] * pred18$p$estimate[1:9] +
         pred18$pmix$estimate[2] * pred18$p$estimate[10:18]
avp21 <- pred21$pmix$estimate[1] * pred21$p$estimate[1:9] +
         pred21$pmix$estimate[2] * pred21$p$estimate[10:18]
avp23 <- pred23$pmix$estimate[1] * pred23$p$estimate[1:9] +
         pred23$pmix$estimate[2] * pred23$p$estimate[10:18]
avphi21 <- pred21$pmix$estimate[1] * pred21$phi$estimate[1:9] +
           pred21$pmix$estimate[2] * pred21$phi$estimate[10:18]
avphi23 <- pred23$pmix$estimate[1] * pred23$phi$estimate[1:9] +
           pred23$pmix$estimate[2] * pred23$phi$estimate[10:18]

xv <- (0:8)+0.5
par(mfrow = c(1,2), xpd = TRUE)
plot(fitsCJS[[17]], 'phi', ylim = c(0,1), type = 'o', lty = 2, CI = FALSE)
lines(xv, pred18$phi$estimate)
lines(xv, avphi21)
lines(xv, avphi23)

plot(fitsCJS[[17]], 'p', ylim = c(0,1), type = 'o', lty = 2, CI = FALSE)
lines(0:8, avp18)
lines(0:8, avp21)
lines(0:8, avp23)
```




As we saw in the initial summary, the number of trapping days per session varied from 4 to 10. Some variation in capture probability is due to varying effort. We compare the collapsed CJS analysis of Pledger et al. (2003) to a full robust-design CJS analysis.

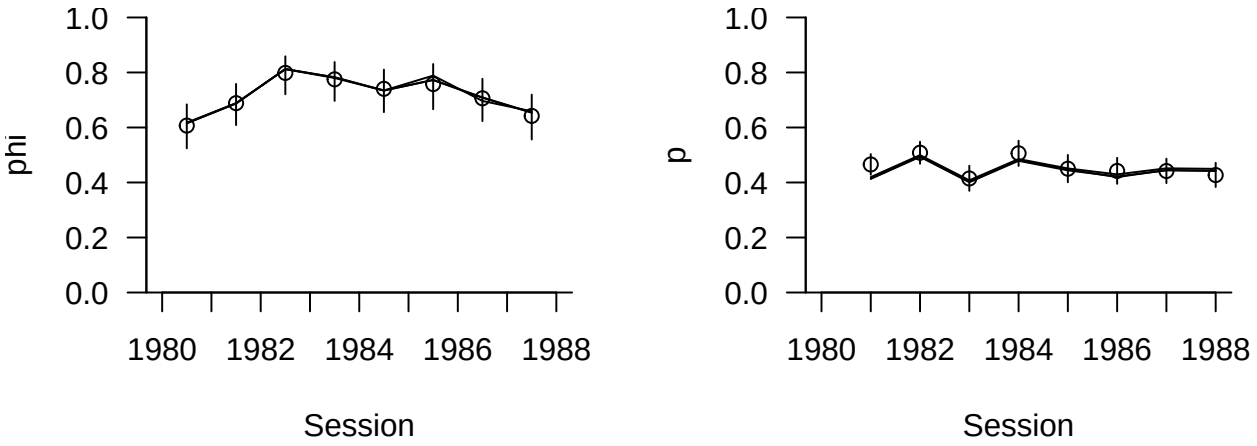
```
baseargs <- list(capthist = 'FebpossumCH', type = 'CJS')
args <- rep(list(baseargs),8)
args[[1]]$model <- list(p~t, phi~t)
args[[2]]$model <- list(p~t+h2, phi~t)
args[[3]]$model <- list(p~t*h2, phi~t)
args[[4]]$model <- list(p~t, phi~t+h2)
args[[5]]$model <- list(p~t+h2, phi~t+h2)
args[[6]]$model <- list(p~t, phi~t*h2)
args[[7]]$model <- list(p~t+h2, phi~t*h2)
args[[8]]$model <- list(p~t*h2, phi~t*h2)
fitsCJSRD <- par.openCR.fit(args, ncores = 8)
```

```
## Warning in par.openCR.fit(args, ncores = 8): par.secr.fit with ncores > 1 is SLOWER than
## par.secr.fit with ncores = 1

## Completed in 1.548 minutes at 12:37:19 17 Jan 2022
```

```
AIC(fitsCJSRD, criterion = 'AIC', sort = FALSE)[,-(5:6)]
```

```
##                          model npar rank logLik    dAIC  AICwt
## 1                    p~t phi~t   16   16  -3541 425.769 0.0000
## 2      p~t + h2 phi~t pmix~h2   18   18  -3328   4.066 0.0908
## 3      p~t * h2 phi~t pmix~h2   25   25  -3323   6.796 0.0232
## 4      p~t phi~t + h2 pmix~h2   18   17  -3521 389.058 0.0000
## 5 p~t + h2 phi~t + h2 pmix~h2   19   19  -3328   5.750 0.0391
## 6      p~t phi~t * h2 pmix~h2   25   18  -3520 400.968 0.0000
## 7 p~t + h2 phi~t * h2 pmix~h2   26   25  -3318   0.000 0.6936
## 8 p~t * h2 phi~t * h2 pmix~h2   33   32  -3313   3.019 0.1533
```

```
average <- function (fit, parm = 'phi') {
    pred <- predict(fit, all = TRUE)
    nr <- nrow(pred[[parm]])
    pred$pmix$estimate[1] * pred[[parm]]$estimate[1:(nr/2)] +
        pred$pmix$estimate[2] * pred[[parm]]$estimate[(nr/2+1):nr]
}
avp2 <- average(fitsCJSRD[[2]], 'p')
avp3 <- average(fitsCJSRD[[3]], 'p')
avp5 <- average(fitsCJSRD[[5]], 'p')
avphi5 <- average(fitsCJSRD[[5]], 'phi')

par(mfrow=c(1,2))
xv <- (0:8)+0.5
plot(fitsCJSRD[[1]], ylim = c(0,1))
plot(fitsCJSRD[[2]], add = TRUE, CI = FALSE, type = 'l')
plot(fitsCJSRD[[3]], add = TRUE, CI = FALSE, type = 'l')
lines(xv, avphi5)
xv <- 0:8
plot(fitsCJSRD[[1]], 'p', ylim = c(0,1))
lines(xv, avp2)
lines(xv, avp3)
lines(xv, avp5)
```



### JSSA analyses

Pledger et al. (2010) analysed data from February samples in 1980–1988, using captures from only the first day of each primary session.

```
FebD1CH <- subset(FebpossumCH, occasion = 1)
JS.counts(FebD1CH)
```

```
##    n  R  m  r  z
## 1 65 65  0 36  0
## 2 78 78 27 53  9
## 3 82 82 44 62 18
## 4 85 85 57 57 23
## 5 76 76 58 46 22
## 6 70 70 54 41 14
## 7 46 46 34 30 21
## 8 78 78 42 22  9
## 9 37 37 31  0  0
```

```
m.array(FebD1CH)
```

```
##       R 1981 1982 1983 1984 1985 1986 1987 1988 NRecap
## 1980 65   27    5    2    1    1    0    0    0     29
## 1981 78        39    8    4    2    0    0    0     25
## 1982 82             47   10    4    0    1    0     20
## 1983 85                  43    9    2    2    1     28
## 1984 76                       38    5    2    1     30
## 1985 70                            27   11    3     29
## 1986 46                                 26    4     16
## 1987 78                                      22     56
## 1988 37                                             37
```

We fit all the models fitted by Pledger et al. (2010)

```
baseargs <- list(capthist = 'FebD1CH', type = 'JSSAb')
args <- rep(list(baseargs),24)
args[[1]]$model <- list(p~1, phi~1, b~t)
args[[2]]$model <- list(p~1, phi~t, b~t)
args[[3]]$model <- list(p~1, phi~h2, b~t)
args[[4]]$model <- list(p~1, phi~t+h2, b~t)
args[[5]]$model <- list(p~1, phi~t*h2, b~t)
args[[6]]$model <- list(p~t, phi~1, b~t)
args[[7]]$model <- list(p~t, phi~t, b~t)
args[[8]]$model <- list(p~t, phi~h2, b~t)
args[[9]]$model <- list(p~t, phi~t+h2, b~t)
args[[10]]$model <- list(p~t, phi~t*h2, b~t)
args[[11]]$model <- list(p~h2, phi~1, b~t)
args[[12]]$model <- list(p~h2, phi~t, b~t)
args[[13]]$model <- list(p~h2, phi~h2, b~t)
args[[14]]$model <- list(p~h2, phi~t+h2, b~t)
args[[15]]$model <- list(p~h2, phi~t*h2, b~t)
args[[16]]$model <- list(p~t+h2, phi~1, b~t)
args[[17]]$model <- list(p~t+h2, phi~t, b~t)
args[[18]]$model <- list(p~t+h2, phi~h2, b~t)
args[[19]]$model <- list(p~t+h2, phi~t+h2, b~t)
args[[20]]$model <- list(p~t+h2, phi~t*h2, b~t)
args[[21]]$model <- list(p~t*h2, phi~1, b~t)
```

```
args[[22]]$model <- list(p~t*h2, phi~t, b~t)
args[[23]]$model <- list(p~t*h2, phi~h2, b~t)
args[[24]]$model <- list(p~t*h2, phi~t+h2, b~t)
fits <- par.openCR.fit(args, ncores = 7)
```

```
## Warning in par.openCR.fit(args, ncores = 7): par.secr.fit with ncores > 1 is SLOWER than
## par.secr.fit with ncores = 1
```

```
## Completed in 2.202 minutes at 12:39:33 17 Jan 2022
```

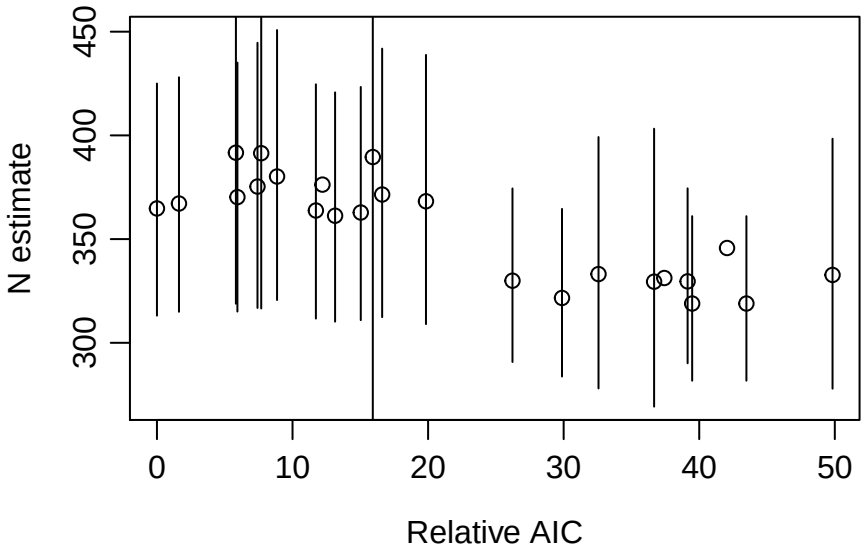
Reproducing Pledger et al. (2010) Table 1.

```
AICtable <- round(matrix(c(AIC(fits, sort = FALSE, criterion = 'AIC')$dAIC,NA), nrow = 5),1)
dimnames(AICtable) <- list(c('phi(.)', 'phi(t)', 'phi(h2)', 'phi(t + h2)', 'phi(t x h2)'),
                           c('p(.)', 'p(t)', 'p(h2)', 'p(t + h2)', 'p(t x h2)'))
AICtable
```

```
##             p(.) p(t) p(h2) p(t + h2) p(t x h2)
## phi(.)      39.5 37.4  13.1       5.9      12.2
## phi(t)      29.9 42.1   0.0       5.8      15.9
## phi(h2)     43.5 32.6  15.0       7.4      11.7
## phi(t + h2) 26.2 36.7   1.6       7.7      16.6
## phi(t x h2) 39.1 49.8   8.9      19.8        NA
```

Reproducing Pledger et al. (2010) Fig. 1.

```
par(mar=c(4,4,2,2), mgp=c(2.4,0.7,0))
Nhat <- sapply(lapply(predict(fits), '[[', 'superN'), '[',1, 'estimate')
Nhatlcl <- sapply(lapply(predict(fits), '[[', 'superN'), '[',1, 'lcl')
Nhatucl <- sapply(lapply(predict(fits), '[[', 'superN'), '[',1, 'ucl')
daic <- AIC(fits, sort = FALSE, criterion = 'AIC')$dAIC
plot(daic, Nhat, ylim = c(270,450), xlab = 'Relative AIC', ylab = 'N estimate')
segments(daic, Nhatlcl, daic, Nhatucl)
```




The superpopulation estimate for the model with lowest AIC was 364.8, compared to 391.7 for the {phi~t, p~t+h2} model.

Reproducing Pledger et al. (2010) Fig. 3.

```
par(mar = c(4,4,2,2))
plot(fits[[17]], 'phi', pch = 16, ylim = c(0,1), CI = FALSE, type = 'o')
phi <- predict(fits[[17]])$phi$estimate
SEphi <- predict(fits[[17]])$phi$SE.estimate
xv <- (0:8)+0.5; segments (xv, phi-SEphi,xv, phi+SEphi)
plot(fits[[17]], 'b',add = TRUE, int = FALSE, CI = FALSE, col = 'red', pch = 16, type = 'o')
abline(h = seq(0,1,0.2), lty = 2)
```

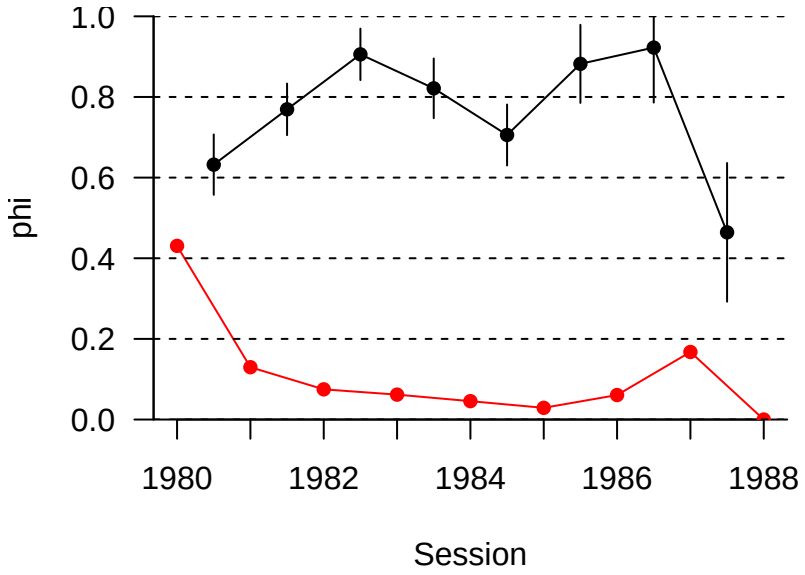



**Fig.** Estimates of apparent survival phi (black) and entry probability b (red) for Orongorongo possums 1980–1988. Bars are +/- 1 SE.

# Spatially explicit analyses

## Kielder Forest field voles

Ergon and Gardner (2014) provided a robust design dataset from a trapping study on field voles *Microtus agrestis* in a clearcut within Kielder Forest, northern England – see also Ergon et al. (2011). The study aimed to describe sex differences in space-use, survival and dispersal among adult voles. Data were from one trapping grid in summer 2000.

Trapping was on a rectangular grid of 192 multi-catch (Ugglan Special) traps at 7-meter spacing. Traps were baited with whole barley grains and carrots; voles were marked with individually numbered ear tags. Traps were checked at about 12 hour intervals at around 6 am and 6 pm such that each primary trapping period had respectively 3, 5, 4 and 5 secondary trap checks. Four primary trapping sessions were begun at intervals of 21 to 23 days between 10 June and 15 August.

Ergon and Gardner (2014) focussed on models for dispersal (movement of home ranges between primary sessions). Modelling dispersal is expected to provide CJS estimates of survival $\phi$ free of the effect of emigration.

### Data summary

```
JS.counts(fieldvoleCH)
```

```
##     n   R  m  r z
## 1  94  91  0 69 0
## 2 106 105 66 88 3
```

```
## 3 109 109 90 89 1
## 4  95  94 90  0 0
```

```
m.array(fieldvoleCH)
```

```
##     R   2   3   4 NRecap
## 1  91  66   2   1     22
## 2 105      88   0     17
## 3 109          89     20
## 4  94                 94
```

Form multi-session capthist for each sex:

```
chF <- subset(fieldvoleCH, function(x) covariates(x)$sex=='F')
chM <- subset(fieldvoleCH, function(x) covariates(x)$sex=='M')
```

First, a conventional CJS analysis on the collapsed data (not robust design). Both parameters p and phi are fitted as constant across sessions.

```
chF1 <- reduce(chF, by = 'all', verify = FALSE)
chM1 <- reduce(chM, by = 'all', verify = FALSE)
CJSfitF <- openCR.fit(chF1)
CJSfitM <- openCR.fit(chM1)
predict(CJSfitF)
```

```
## $p
##   session estimate SE.estimate    lcl    ucl
## 1       1       NA          NA     NA     NA
## 2       2   0.9887     0.01113 0.9253 0.9984
## 3       3   0.9887     0.01113 0.9253 0.9984
## 4       4   0.9887     0.01113 0.9253 0.9984
##
## $phi
##   session estimate SE.estimate    lcl    ucl
## 1       1   0.7174     0.03488 0.6444 0.7806
## 2       2   0.7174     0.03488 0.6444 0.7806
## 3       3   0.7174     0.03488 0.6444 0.7806
## 4       4       NA          NA     NA     NA
```

```
predict(CJSfitM)
```

```
## $p
##   session estimate SE.estimate    lcl    ucl
## 1       1       NA          NA     NA     NA
## 2       2   0.9341     0.03545 0.8209 0.9777
## 3       3   0.9341     0.03545 0.8209 0.9777
## 4       4   0.9341     0.03545 0.8209 0.9777
##
## $phi
##   session estimate SE.estimate    lcl    ucl
## 1       1   0.8775     0.04542 0.7578 0.9425
## 2       2   0.8775     0.04542 0.7578 0.9425
## 3       3   0.8775     0.04542 0.7578 0.9425
## 4       4       NA          NA     NA     NA
```

These estimates match the nonspatial MCMC results reported by Ergon and Gardner (2014) in their Table 4 (section B).

Next, try a spatial analysis on the robust-design data. Ergon and Gardner (2014) appear to have used a 14-m buffer; this is rather small but we use it here to be consistent. In detection function 'HVP' the parameter z matches their exponent $\kappa$ ($\kappa = 1$ corresponds to 'HEX' and $\kappa = 2$ to 'HHN').

```
msk <- make.mask(traps(chF[[1]]), buffer = 14, spacing = 1, type = 'traprect')
ampmdf <- attr(fieldvoleCH, 'ampm')
baseargs <- list(mask = msk, type = 'CJSsecr',  detectfn = 'HVP',
    model = list(lambda0~ampm, phi~1, sigma~1, z~1), timecov = ampmdf,
    movementmodel = 'RDE', details = list(CJSp1 = TRUE))
args <- rep(list(baseargs),2)
args[[1]]$capthist <- 'chF'
args[[2]]$capthist <- 'chM'
fits <- par.openCR.fit(args)
```

```
## Completed in 29.105 minutes at 13:08:53 17 Jan 2022
```

```
names(fits) <- c('exponF', 'exponM')
```

We could view estimates of all 'real' parameters for all models with

```
predict(fits, all.levels = TRUE)
```

A more simple summary is

```
make.table(fits, 'phi')
```

```
##         session
## model         1      2      3 4
##   exponF 0.7533 0.7533 0.7533
##   exponM 0.8897 0.8897 0.8897
```

```
make.table(fits, 'lambda0')
```

```
##         session
## model         1      2      3      4
##   exponF 3.9386 3.9386 3.9386 3.9386
##   exponM 0.4311 0.4311 0.4311 0.4311
```

```
make.table(fits, 'sigma')
```

```
##         session
## model        1     2     3     4
##   exponF 1.134 1.134 1.134 1.134
##   exponM 8.842 8.842 8.842 8.842
```

```
make.table(fits, 'z')
```

```
##         session
## model         1      2      3      4
##   exponF 0.7115 0.7115 0.7115 0.7115
##   exponM 1.6215 1.6215 1.6215 1.6215
```

```
make.table(fits, 'move.a')
```

```
##         session
## model        1     2     3 4
##   exponF 3.018 3.018 3.018
##   exponM 5.305 5.305 5.305
```

These results differ somewhat from those of Ergon and Gardner (2014) Table 4.

```
baseargs <- list(mask = msk, type = 'CJSsecr',  detectfn = 'HVP',
    model = list(lambda0~ampm, phi~1, sigma~1, z~1), timecov = ampmdf,
    movementmodel = 'INDzi', details = list(CJSp1 = TRUE))
args <- rep(list(baseargs),2)
args[[1]]$capthist <- 'chF'
args[[2]]$capthist <- 'chM'
fitszi <- par.openCR.fit(args)
```

```
## Completed in 11.343 minutes at 13:20:25 17 Jan 2022
```

```
names(fitszi) <- c('exponF', 'exponM')
```

```
baseargs <- list(mask = msk, type = 'PLBsecrf',  detectfn = 'HVP',
    model = list(lambda0~ampm, phi~1, sigma~1, z~1), timecov = ampmdf,
    movementmodel = 'INDzi')
args <- rep(list(baseargs),2)
args[[1]]$capthist <- 'chF'
args[[2]]$capthist <- 'chM'
fitsziPLB <- par.openCR.fit(args)
```

```
## Completed in 29.381 minutes at 13:49:50 17 Jan 2022
```

```
names(fitsziPLB) <- c('exponF', 'exponM')
```

```
msk <- make.mask(traps(chF), buffer = 21, spacing = 1, type = 'traprect')
baseargs <- list(mask = msk, type = 'CJSsecr',  detectfn = 'HVP',
    model = list(lambda0~ampm, phi~1, sigma~1, z~1), timecov = ampmdf,
    movementmodel = 'INDzi', details = list(CJSp1 = TRUE))
args <- rep(list(baseargs),2)
args[[1]]$capthist <- 'chF'
args[[2]]$capthist <- 'chM'
fitszi21 <- par.openCR.fit(args)
```

```
## Warning in openCR.fit(mask = structure(list(`1` = structure(list(x = c(-20.5, : multi-
## session mask provided; using first
```

```
## Warning in openCR.fit(mask = structure(list(`1` = structure(list(x = c(-20.5, : multi-
## session mask provided; using first
```

```
## Completed in 14.971 minutes at 14:04:50 17 Jan 2022
```

```
names(fitszi21) <- c('exponF', 'exponM')
```

```
msk <- make.mask(traps(chF), buffer = 14, spacing = 1, type = 'traprect')
baseargs <- list(mask = msk, type = 'CJSsecr',  detectfn = 'HVP',
    model = list(lambda0~ampm, phi~1, sigma~1, z~1), timecov = ampmdf,
    movementmodel = 'RDEzi', details = list(CJSp1 = TRUE))
args <- rep(list(baseargs),2)
args[[1]]$capthist <- 'chF'
args[[2]]$capthist <- 'chM'
fitsfrEzi <- par.openCR.fit(args)
```

```
## Warning in openCR.fit(mask = structure(list(`1` = structure(list(x = c(-13.5, : multi-
## session mask provided; using first
```

```
## Warning in openCR.fit(mask = structure(list(`1` = structure(list(x = c(-13.5, : multi-
## session mask provided; using first
```

```
## Completed in 37.816 minutes at 14:42:43 17 Jan 2022
```

```
names(fitsfrEzi) <- c('exponF', 'exponM')
```

```
fitfnorm <- openCR.fit(chF, mask = msk, type = 'PLBsecrf',  detectfn = 'HHN',
           model = list(lambda0~ampm, phi~1, f~1, sigma~1), timecov = ampmdf,
           movementmodel = 'BVN')
```

```
## Warning in openCR.fit(chF, mask = msk, type = "PLBsecrf", detectfn = "HHN", : multi-
## session mask provided; using first
```

```
fitmnorm <- openCR.fit(chM, mask = msk, type = 'PLBsecrf',  detectfn = 'HHN',
           model = list(lambda0~ampm, phi~1, f~1, sigma~1), timecov = ampmdf,
           movementmodel = 'BVN')
```

```
## Warning in openCR.fit(chM, mask = msk, type = "PLBsecrf", detectfn = "HHN", : multi-
## session mask provided; using first
```

```
fitm <- openCR.fit(chM, type = 'PLBf',  detectfn = 'HHN',
           model = list(lambda0~ampm, phi~1, f~1, sigma~1), timecov = ampmdf)
fits <- openCRlist(fitfnorm, fitmnorm)
make.table(fits, 'phi')
```

```
##           session
## model           1      2      3 4
##   fitfnorm 0.7333 0.7333 0.7333
##   fitmnorm 0.8960 0.8960 0.8960
```

```
make.table(fits, 'sigma')
```

```
##           session
## model          1     2     3     4
##   fitfnorm 5.338 5.338 5.338 5.338
##   fitmnorm 7.259 7.259 7.259 7.259
```

```
make.table(fits, 'move.a')
```

```
##           session
## model          1     2     3 4
##   fitfnorm 2.465 2.465 2.465
##   fitmnorm 4.020 4.020 4.020
```

```
make.table(fits, 'move.a')/make.table(fits, 'sigma')
```

```
##           session
## model           1      2      3 4
##   fitfnorm 0.4618 0.4618 0.4618
##   fitmnorm 0.5537 0.5537 0.5537
```

## Patuxent ovenbirds

Ovenbirds (*Seiurus aurocapilla*) were captured in 44 mist nets on 9–10 days during the breeding season in 5 consecutive years. Each received a uniquely numbered metal band. The **secr** capthist object `ovenCHp` uses the 'proximity' detector type: capture of an individual in a particular net on a particular day was a binary variable.

### Data summary

```
summary(ovenCHp, terse = TRUE)
```

```
##            2005 2006 2007 2008 2009
## Occasions     9   10   10   10   10
## Detections   41   49   57   33   35
## Animals      20   22   26   19   16
## Detectors    44   44   44   44   44
```

```
m.array(ovenCHp)
```

```
##       R 2006 2007 2008 2009 NRecap
## 2005 20    9    2    1    0      8
## 2006 22        10    0    0     12
## 2007 26              3    3     20
## 2008 19                   5     14
## 2009 15                         15
```

### Non-spatial analysis

Suppose we are interested in the overall population trend represented by the realised finite rate of increase ($\lambda$).

```
baseargs <- list(capthist = 'ovenCH')
args <- rep(list(baseargs), 10)
names(args) <- c('cjs', 'plbb', 'plbf', 'plbg', 'plbl',
                 'jssab', 'jssaf', 'jssag', 'jssal', 'jssaN')
args[[1]] <- c(args[[1]], list(type = 'CJS',     model = list(p~t, phi~t)))
args[[2]] <- c(args[[2]], list(type = 'PLBb', model = list(p~t, phi~t, b~t)))
args[[3]] <- c(args[[3]], list(type = 'PLBf', model = list(p~t, phi~t, f~t)))
args[[4]] <- c(args[[4]], list(type = 'PLBg', model = list(p~t, phi~t, gamma~t)))
args[[5]] <- c(args[[5]], list(type = 'PLBl', model = list(p~t, phi~t, lambda~t)))
args[[6]] <- c(args[[6]], list(type = 'JSSAb',   model = list(p~t, phi~t, b~t)))
args[[7]] <- c(args[[7]], list(type = 'JSSAf',   model = list(p~t, phi~t, f~t)))
args[[8]] <- c(args[[8]], list(type = 'JSSAg',   model = list(p~t, phi~t, gamma~t)))
args[[9]] <- c(args[[9]], list(type = 'JSSAl',   model = list(p~t, phi~t, lambda~t)))
args[[10]] <- c(args[[10]], list(type = 'JSSAN',   model = list(p~t, phi~t, N~t)))
fits <- par.openCR.fit(args, ncores = 5)
```

```
## Completed in 0.727 minutes at 15:22:09 17 Jan 2022
```

```
make.table(fits, parm = "p")
```

```
##        session
## model      2005    2006    2007    2008    2009
##   cjs           0.16890 0.22441 0.04917 0.18504
##   plbb  0.14886 0.13841 0.16420 0.08566 0.17676
##   plbf  0.14886 0.13841 0.16420 0.08566 0.17676
##   plbg  0.14886 0.13841 0.16420 0.08566 0.17676
##   plbl  0.14886 0.13841 0.16420 0.08566 0.17676
##   jssab 0.14886 0.13841 0.16420 0.08566 0.17676
##   jssaf 0.14886 0.13841 0.16420 0.08566 0.17676
##   jssag 0.14886 0.13841 0.16420 0.08566 0.17676
##   jssal 0.14886 0.13841 0.16420 0.08566 0.17676
##   jssaN 0.14886 0.13841 0.16420 0.08566 0.17676
```

```
make.table(fits, parm = "phi")
```

```
##        session
## model     2005   2006   2007   2008 2009
```

```
##   cjs   0.6512 0.5308 0.4387 0.3377
##   plbb  0.6780 0.5600 0.3348 0.3784
##   plbf  0.6780 0.5600 0.3348 0.3784
##   plbg  0.6780 0.5600 0.3348 0.3784
##   plbl  0.6780 0.5600 0.3348 0.3784
##   jssab 0.6780 0.5600 0.3348 0.3784
##   jssaf 0.6780 0.5600 0.3348 0.3784
##   jssag 0.6780 0.5600 0.3348 0.3784
##   jssal 0.6780 0.5600 0.3348 0.3784
##   jssaN 0.6780 0.5600 0.3348 0.3784
```

All parameterizations of the JSSA model yield exactly the same estimates of p and phi.

For later reference, fit a nonspatial model with constant p and lambda:

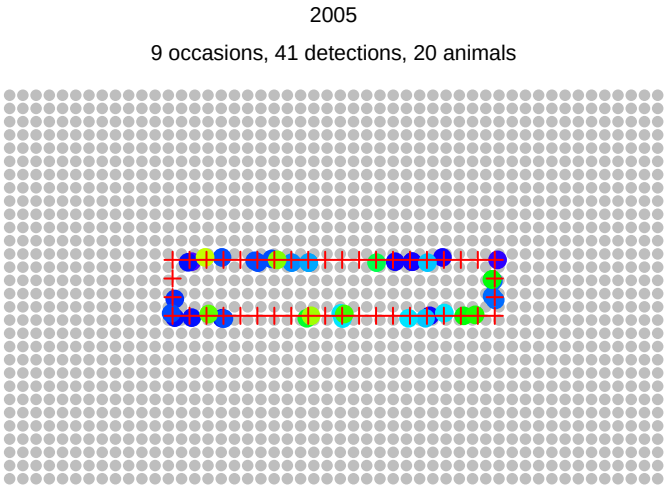
```
fitns <- openCR.fit(ovenCHp, type = 'PLBl', model = phi ~ t)
predict(fitns)
```

```
## $p
##   session estimate SE.estimate    lcl    ucl
## 1    2005   0.1411      0.0124 0.1185 0.1672
## 2    2006   0.1411      0.0124 0.1185 0.1672
## 3    2007   0.1411      0.0124 0.1185 0.1672
## 4    2008   0.1411      0.0124 0.1185 0.1672
## 5    2009   0.1411      0.0124 0.1185 0.1672
##
## $phi
##   session estimate SE.estimate    lcl    ucl
## 1    2005   0.6327     0.10943 0.4063 0.8126
## 2    2006   0.5178     0.09646 0.3349 0.6960
## 3    2007   0.2815     0.08789 0.1432 0.4787
## 4    2008   0.5149     0.12646 0.2824 0.7412
## 5    2009       NA          NA     NA     NA
##
## $lambda
##   session estimate SE.estimate    lcl   ucl
## 1    2005   0.9551     0.06968 0.8279 1.102
## 2    2006   0.9551     0.06968 0.8279 1.102
## 3    2007   0.9551     0.06968 0.8279 1.102
## 4    2008   0.9551     0.06968 0.8279 1.102
## 5    2009       NA          NA     NA    NA
```

### Spatial analysis

For spatial analyses we need to define a habitat mask; this represents the extent of relevant habitat. We use the (default) mask type 'traprect' to allow toroidal wrapping. The mask type 'trapbuffer' was used previously, but that now requires the slower edge method 'truncate' (see Edge effects in the openCR vignette).

```
par(mar = c(2,2,2,2))
ovenCHr <- rotate(ovenCHp, 90)  # rotate for horizontal plot
nets <- traps(ovenCHr)[[1]]
msk <- make.mask(nets, buffer = 300, nx = 50, type = "traprect")
plot(msk)
plot(ovenCHr[[1]], add = TRUE)
plot(nets, tracks = TRUE, add = TRUE)
```




Now we can perform a spatial analysis equivalent to fitns. This takes much longer to fit.

```
msk <- make.mask(traps(ovenCHp[[1]]), buffer = 300, nx = 25, type = "traprect")
fitsp <- openCR.fit(ovenCHp, type = 'PLBsecrl', model = lambda0~t, mask = msk)
```

```
predict(fitsp)
```

```
## $lambda0
##   session estimate SE.estimate     lcl     ucl
## 1    2005  0.01972    0.004374 0.01277 0.03046
## 2    2006  0.02079    0.004013 0.01424 0.03035
## 3    2007  0.02986    0.005971 0.02018 0.04419
## 4    2008  0.01656    0.004230 0.01004 0.02732
## 5    2009  0.02212    0.006451 0.01249 0.03918
##
## $phi
##   session estimate SE.estimate    lcl    ucl
## 1    2005   0.5325     0.06203 0.4114 0.6498
## 2    2006   0.5325     0.06203 0.4114 0.6498
## 3    2007   0.5325     0.06203 0.4114 0.6498
## 4    2008   0.5325     0.06203 0.4114 0.6498
## 5    2009       NA          NA     NA     NA
##
## $lambda
##   session estimate SE.estimate    lcl   ucl
## 1    2005   0.9683     0.07759 0.8276 1.133
## 2    2006   0.9683     0.07759 0.8276 1.133
## 3    2007   0.9683     0.07759 0.8276 1.133
## 4    2008   0.9683     0.07759 0.8276 1.133
## 5    2009       NA          NA     NA    NA
##
## $sigma
##   session estimate SE.estimate   lcl ucl
## 1    2005    99.44       7.371 85.99 115
## 2    2006    99.44       7.371 85.99 115
## 3    2007    99.44       7.371 85.99 115
## 4    2008    99.44       7.371 85.99 115
## 5    2009    99.44       7.371 85.99 115
```

Notice that the estimate of sigma is larger than for models fitted to the succession of closed populations using `secr.fit`. For example

```
collate(ovenbird.model.1)[,,,'sigma']
```

```
##              estimate SE.estimate   lcl   ucl
## session=2005    78.58       6.386 67.02 92.12
## session=2006    78.58       6.386 67.02 92.12
## session=2007    78.58       6.386 67.02 92.12
## session=2008    78.58       6.386 67.02 92.12
## session=2009    78.58       6.386 67.02 92.12
```

We attribute this to year-to-year shifts in activity centre that inflate sigma for the open population model. Next we fit a model that allows birds to shift their home ranges from year to year using either a bivariate normal (BVN) or bivariate Laplacian (BVE) kernel. We can save a little time by starting at the estimates from the previous model.

```
fitsp2 <- openCR.fit(ovenCHp, type = 'PLBsecrl', model = lambda0~t, mask = msk,
                     movementmodel = 'BVN')
```

```
fitsp3 <- openCR.fit(ovenCHp, type = 'PLBsecrl', model = lambda0~t, mask = msk,
                     movementmodel = 'BVE')
```

```
ocr <- openCRlist(static = fitsp, BVN = fitsp2, BVE = fitsp3)
AIC(ocr)[,-1]   # drop 'model' column to save space
```

```
##        npar rank logLik  AIC AICc   dAIC  AICwt
## BVE       9    9  -1430 2877 2880  0.000 0.8746
## BVN       9    9  -1431 2881 2884  3.884 0.1254
## static    8    8  -1446 2908 2910 30.988 0.0000
```

Modelling between-year movement improved the fit, and all parameters could be estimated (rank equal to number of parameters). We compare estimates of the within-session detection parameters:

```
make.table(ocr, 'lambda0')
```

```
##         session
## model       2005    2006    2007    2008    2009
##   static 0.01972 0.02079 0.02986 0.01656 0.02212
##   BVN    0.03305 0.03241 0.03668 0.02114 0.02614
##   BVE    0.03266 0.03203 0.03798 0.02154 0.02671
```

```
make.table(ocr, 'lambda0', 'SE.estimate')
```

```
##         session
## model        2005     2006     2007     2008     2009
##   static 0.004374 0.004013 0.005971 0.004230 0.006451
##   BVN    0.008086 0.007046 0.007439 0.005472 0.007221
##   BVE    0.007892 0.006820 0.007732 0.005567 0.007372
```

```
make.table(ocr, 'sigma')
```

```
##         session
## model     2005  2006  2007  2008  2009
##   static 99.44 99.44 99.44 99.44 99.44
##   BVN    76.26 76.26 76.26 76.26 76.26
##   BVE    75.33 75.33 75.33 75.33 75.33
```

```
make.table(ocr, 'sigma', 'SE.estimate')
```

```
##         session
## model     2005  2006  2007  2008  2009
##   static 7.371 7.371 7.371 7.371 7.371
##   BVN    5.524 5.524 5.524 5.524 5.524
##   BVE    5.382 5.382 5.382 5.382 5.382
```

The estimate of sigma is now in line with that from the closed-population models. Interestingly, the constant survival rate has increased considerably. It would be rash to claim this fully accounts for the emigration component of apparent survival, but it may be getting close.

```
make.table(ocr, 'phi')
```

```
##         session
## model      2005   2006   2007   2008 2009
##   static 0.5325 0.5325 0.5325 0.5325
##   BVN    0.6218 0.6218 0.6218 0.6218
##   BVE    0.6276 0.6276 0.6276 0.6276
```

```
make.table(ocr, 'phi', 'SE.estimate')
```

```
##         session
## model       2005    2006    2007    2008 2009
##   static 0.06203 0.06203 0.06203 0.06203
##   BVN    0.07455 0.07455 0.07455 0.07455
##   BVE    0.07657 0.07657 0.07657 0.07657
```

For a final flourish, compare all estimates of population growth rate with an estimate from **secr**, using estimated trend over a set of closed-population samples.

```
ocr2 <- openCRlist(static = fitsp, BVN = fitsp2, BVE = fitsp3)
# from open-population models
do.call(rbind, lapply(ocr2, function(x) predict(x)$lambda[1, c('estimate','lcl','ucl')]))
```

```
##        estimate    lcl   ucl
## static   0.9683 0.8276 1.133
## BVN      0.9943 0.8507 1.162
## BVE      0.9918 0.8489 1.159
```

```
# from 'secr' pre-fitted multisession closed population model D ~ Session
# exponentiate the 'beta' coefficient
exp(coef(ovenbird.model.D)['D.Session',-2])
```

```
##             beta    lcl   ucl
## D.Session 0.9381 0.8176 1.076
```

## Orongorongo Valley possums robust design

Brushtail possums *Trichosurus vulpecula* were trapped from 1980–2006 according to a robust-design scheme with three primary sessions per year (nominally February, June and September). There is a single annual birth pulse in May; young are carried in the mother's pouch through about October-November and ride on the mother's back for some weeks after leaving the pouch. The three samples therefore correspond to post-independence (February), early pouch phase (June) and late pouch phase (September).

The dataset OVpossumCH in **secr** includes data for the six sessions in 1996 and 1997 (sessions 49–54). The population was at high density at this time.

### Data summary

```
summary(OVpossumCH, terse = TRUE)
```

```
##             49  50  51  52  53  54
## Occasions    5   5   5   5   5   5
## Detections 450 494 328 383 372 375
## Animals    223 206 148 162 154 135
## Detectors  167 167 167 167 167 167
```

```
m.array(OVpossumCH)
```

```
##      R  50  51  52  53  54 NRecap
## 49 220 167   9   4   3   0     37
## 50 203     130  11   3   0     59
## 51 147         124   3   0     20
## 52 162             135   4     23
## 53 154                 126     28
## 54 134                       134
```

### Analysis

Construct a habitat mask, using a forest map to exclude riverbed.

```
ovtrap <- traps(OVpossumCH[[1]])
if (!requireNamespace('rgdal')) stop ("this example requires package rgdal")
```

```
## Loading required namespace: rgdal
```

```
datadir <- system.file("extdata", package = "secr")
OVforest <- rgdal::readOGR(dsn = datadir, layer = "OVforest")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\R\R-4.0.5\library\secr\extdata", layer: "OVforest"
## with 3 features
## It has 2 fields
```

```
ovmask <- make.mask(ovtrap, buffer = 120, type = 'trapbuffer',
    poly = OVforest[1:2,], spacing = 15, keep.poly = FALSE)
ovmask2 <- make.mask(ovtrap, buffer = 200, type = 'trapbuffer',
    poly = OVforest[1:2,], spacing = 15, keep.poly = FALSE)
intervals(OVpossumCH) <- c(123, 91, 154, 118, 85)/365  # annual
```

Fit three models, two with season as a covariate. Intervals were not included in the **secr** dataset, so we sneak them in here. Sexes and ages are pooled.

```
baseargs <- list(capthist = 'OVpossumCH', mask = ovmask, type = 'PLBsecrf',
                 sessioncov = c(1,2,3,1,2,3))
args <- rep(list(baseargs), 3)
names(args) <- c('null','seasonal.phi.f','seasonal.lambda0.phi.f')
args[[1]]$model <- list(lambda0~1, phi~1, f~1)
args[[2]]$model <- list(lambda0~1, phi~scov, f~scov)
args[[3]]$model <- list(lambda0~scov, phi~scov, f~scov)
fits <- par.openCR.fit(args, ncores = 3)
```

```
## Warning in par.openCR.fit(args, ncores = 3): par.secr.fit with ncores > 1 is SLOWER than
## par.secr.fit with ncores = 1
```

```
## Completed in 10.149 minutes at 16:00:25 17 Jan 2022
```

```
AIC(fits)[,-1]  # drop 'model' column to save space
```

```
##                         npar rank logLik   AIC  AICc  dAIC  AICwt
## seasonal.lambda0.phi.f     7    7 -12460 24934 24935 0.000 0.6430
## null                       4    4 -12464 24936 24936 1.434 0.3139
## seasonal.phi.f             6    6 -12464 24940 24940 5.408 0.0430
```

```
predict(fits)
```

```
## $null
## $null$lambda0
##   session estimate SE.estimate     lcl     ucl
## 1      49   0.0812    0.002531 0.07639 0.08632
## 2      50   0.0812    0.002531 0.07639 0.08632
## 3      51   0.0812    0.002531 0.07639 0.08632
## 4      52   0.0812    0.002531 0.07639 0.08632
## 5      53   0.0812    0.002531 0.07639 0.08632
## 6      54   0.0812    0.002531 0.07639 0.08632
##
## $null$phi
##   session estimate SE.estimate    lcl    ucl
## 1      49   0.6235     0.02744 0.5684 0.6756
## 2      50   0.6235     0.02744 0.5684 0.6756
## 3      51   0.6235     0.02744 0.5684 0.6756
## 4      52   0.6235     0.02744 0.5684 0.6756
## 5      53   0.6235     0.02744 0.5684 0.6756
## 6      54       NA          NA     NA     NA
##
## $null$f
##   session estimate SE.estimate    lcl    ucl
## 1      49   0.1043     0.02002 0.0716 0.1519
## 2      50   0.1043     0.02002 0.0716 0.1519
## 3      51   0.1043     0.02002 0.0716 0.1519
## 4      52   0.1043     0.02002 0.0716 0.1519
## 5      53   0.1043     0.02002 0.0716 0.1519
## 6      54       NA          NA     NA     NA
##
## $null$sigma
##   session estimate SE.estimate   lcl   ucl
## 1      49    36.13      0.4428 35.27 37.01
## 2      50    36.13      0.4428 35.27 37.01
## 3      51    36.13      0.4428 35.27 37.01
## 4      52    36.13      0.4428 35.27 37.01
## 5      53    36.13      0.4428 35.27 37.01
## 6      54    36.13      0.4428 35.27 37.01
##
##
## $seasonal.phi.f
## $seasonal.phi.f$lambda0
##   session scov estimate SE.estimate    lcl     ucl
## 1      49    1  0.08123    0.002539 0.0764 0.08636
## 2      50    2  0.08123    0.002539 0.0764 0.08636
## 3      51    3  0.08123    0.002539 0.0764 0.08636
## 4      52    1  0.08123    0.002539 0.0764 0.08636
```

```
## 5      53    2  0.08123    0.002539 0.0764 0.08636
## 6      54    3  0.08123    0.002539 0.0764 0.08636
##
## $seasonal.phi.f$phi
##   session scov estimate SE.estimate    lcl    ucl
## 1      49    1   0.6205     0.04092 0.5377 0.6968
## 2      50    2   0.6241     0.02819 0.5675 0.6776
## 3      51    3   0.6277     0.05105 0.5236 0.7212
## 4      52    1   0.6205     0.04092 0.5377 0.6968
## 5      53    2   0.6241     0.02819 0.5675 0.6776
## 6      54    3       NA          NA     NA     NA
##
## $seasonal.phi.f$f
##   session scov estimate SE.estimate     lcl    ucl
## 1      49    1   0.1076     0.03453 0.05737 0.2018
## 2      50    2   0.1043     0.02002 0.07158 0.1519
## 3      51    3   0.1011     0.03496 0.05131 0.1991
## 4      52    1   0.1076     0.03453 0.05737 0.2018
## 5      53    2   0.1043     0.02002 0.07158 0.1519
## 6      54    3       NA          NA      NA     NA
##
## $seasonal.phi.f$sigma
##   session scov estimate SE.estimate   lcl   ucl
## 1      49    1    36.13      0.4428 35.27 37.01
## 2      50    2    36.13      0.4428 35.27 37.01
## 3      51    3    36.13      0.4428 35.27 37.01
## 4      52    1    36.13      0.4428 35.27 37.01
## 5      53    2    36.13      0.4428 35.27 37.01
## 6      54    3    36.13      0.4428 35.27 37.01
##
##
## $seasonal.lambda0.phi.f
## $seasonal.lambda0.phi.f$lambda0
##   session scov estimate SE.estimate     lcl     ucl
## 1      49    1  0.07552    0.003149 0.06959 0.08195
## 2      50    2  0.08152    0.002552 0.07666 0.08668
## 3      51    3  0.08799    0.003735 0.08096 0.09562
## 4      52    1  0.07552    0.003149 0.06959 0.08195
## 5      53    2  0.08152    0.002552 0.07666 0.08668
## 6      54    3  0.08799    0.003735 0.08096 0.09562
##
## $seasonal.lambda0.phi.f$phi
##   session scov estimate SE.estimate    lcl    ucl
## 1      49    1   0.6136     0.04091 0.5310 0.6901
## 2      50    2   0.6227     0.02821 0.5660 0.6762
## 3      51    3   0.6317     0.05090 0.5277 0.7248
## 4      52    1   0.6136     0.04091 0.5310 0.6901
## 5      53    2   0.6227     0.02821 0.5660 0.6762
## 6      54    3       NA          NA     NA     NA
##
## $seasonal.lambda0.phi.f$f
##   session scov estimate SE.estimate     lcl    ucl
## 1      49    1  0.08582     0.03376 0.03970 0.1855
## 2      50    2  0.09753     0.02014 0.06506 0.1462
```

```
## 3      51    3  0.11083     0.03825 0.05635 0.2180
## 4      52    1  0.08582     0.03376 0.03970 0.1855
## 5      53    2  0.09753     0.02014 0.06506 0.1462
## 6      54    3       NA          NA      NA     NA
##
## $seasonal.lambda0.phi.f$sigma
##   session scov estimate SE.estimate   lcl ucl
## 1      49    1    36.12      0.4426 35.26  37
## 2      50    2    36.12      0.4426 35.26  37
## 3      51    3    36.12      0.4426 35.26  37
## 4      52    1    36.12      0.4426 35.26  37
## 5      53    2    36.12      0.4426 35.26  37
## 6      54    3    36.12      0.4426 35.26  37
```

Using larger mask (200-m buffer) and including movement...

```
nonspOV <- openCR.fit(OVpossumCH, type = 'PLBf',
    model = list(p~t, phi~t, f~t))
spOV <- openCR.fit(OVpossumCH, type = 'PLBsecrf', start = nonspOV,
    model = list(lambda0~t, phi~t, f~t), mask = ovmask2)
# use default start here as 'spOV' gives bad maximization
spmOV <- openCR.fit(OVpossumCH, type = 'PLBsecrf',
    model = list(lambda0~t, phi~t, f~t, move.a~t),
    movementmodel = 'BVN', mask = ovmask2)
save(nonspOV, spOV, spmOV, file = paste0(testdir, 'spOV.RData'))
```

```
load(paste0(testdir, 'spOV.RData'))
# format tables of results
formatOV <- function(ocrlist = list(nonspOV, spOV, spmOV), parm = 'phi', fmt = '%5.3f') {
    formatest <- function () {
        est <- sprintf(fmt, t(make.table(ocrlist, parm = parm, fields = 'estimate')))
        SE <- sprintf(fmt, t(make.table(ocrlist, parm = parm, fields = 'SE.estimate')))
        SE <- paste0 ('(', SE, ')')
        paste(est, SE)
    }
    m1 <- matrix(t(apply(matrix(formatest(), ncol = 3),1,paste, collapse = ' & ')), ncol = 1)
    m2 <- cbind(c('1996','    ','    ','1997','    ','    '), rep(c('Feb','Jun','Sep'),2),
                1:6, m1)
    m3 <- apply(m2, 1, paste, collapse = ' & ')
    out <- sapply(m3, cat, '\\\\ \n')
}
formatOV(parm = 'phi')
```

```
## 1996 & Feb & 1 & 0.606 (0.055) & 0.619 (0.056) & 0.617 (0.057) \\
##      & Jun & 2 & 0.310 (0.051) & 0.331 (0.053) & 0.336 (0.054) \\
##      & Sep & 3 & 0.734 (0.054) & 0.771 (0.052) & 0.776 (0.051) \\
## 1997 & Feb & 4 & 0.655 (0.063) & 0.811 (0.060) & 0.809 (0.060) \\
##      & Jun & 5 & 0.456 (0.075) & 0.617 (0.092) & 0.624 (0.094) \\
##      & Sep & 6 &    NA (   NA) &    NA (   NA) &    NA (   NA) \\
```

```
formatOV(parm = 'f')
```

```
## 1996 & Feb & 1 & 0.175 (0.081) & 0.150 (0.072) & 0.131 (0.071) \\
##      & Jun & 2 & 0.063 (0.029) & 0.032 (0.021) & 0.038 (0.024) \\
##      & Sep & 3 & 0.287 (0.073) & 0.173 (0.060) & 0.160 (0.059) \\
## 1997 & Feb & 4 & 0.127 (0.050) & 0.026 (0.032) & 0.027 (0.034) \\
```

```
##       & Jun & 5 & 0.070 (0.037) & 0.022 (0.022) & 0.018 (0.020) \\
##       & Sep & 6 &    NA (   NA) &    NA (   NA) &    NA (   NA) \\
```

```
formatOV(parm = 'lambda0', fmt = '%5.3f')
```

```
## 1996 & Feb & 1 &    NA (   NA) & 0.062 (0.004) & 0.079 (0.005) \\
##       & Jun & 2 &    NA (   NA) & 0.081 (0.004) & 0.103 (0.006) \\
##       & Sep & 3 &    NA (   NA) & 0.069 (0.004) & 0.089 (0.006) \\
## 1997 & Feb & 4 &    NA (   NA) & 0.091 (0.005) & 0.114 (0.007) \\
##       & Jun & 5 &    NA (   NA) & 0.094 (0.005) & 0.116 (0.007) \\
##       & Sep & 6 &    NA (   NA) & 0.113 (0.007) & 0.142 (0.009) \\
```

```
formatOV(parm = 'sigma', fmt = '%4.1f')
```

```
## 1996 & Feb & 1 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
##       & Jun & 2 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
##       & Sep & 3 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
## 1997 & Feb & 4 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
##       & Jun & 5 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
##       & Sep & 6 &   NA (  NA) & 36.1 ( 0.4) & 31.8 ( 0.5) \\
```

```
formatOV(parm = 'move.a', fmt = '%4.1f')
```

```
## 1996 & Feb & 1 &   NA (  NA) &   NA (  NA) & 20.7 ( 2.5) \\
##       & Jun & 2 &   NA (  NA) &   NA (  NA) & 17.6 ( 2.6) \\
##       & Sep & 3 &   NA (  NA) &   NA (  NA) & 26.8 ( 2.5) \\
## 1997 & Feb & 4 &   NA (  NA) &   NA (  NA) & 17.4 ( 2.3) \\
##       & Jun & 5 &   NA (  NA) &   NA (  NA) & 16.9 ( 2.3) \\
##       & Sep & 6 &   NA (  NA) &   NA (  NA) &   NA (  NA) \\
```

```
make.table(list(nonspOV, spOV, spmOV), 'p')
```

```
##        session
## model       49     50     51     52     53     54
##   fits1 0.3604 0.4349 0.3819 0.4417 0.4660 0.5447
##   fits2
##   fits3
```

```
make.table(list(nonspOV, spOV, spmOV), 'sigma')
```

```
##        session
## model      49    50    51    52    53    54
##   fits1
##   fits2 36.13 36.13 36.13 36.13 36.13 36.13
##   fits3 31.81 31.81 31.81 31.81 31.81 31.81
```

```
make.table(list(nonspOV, spOV, spmOV), 'sigma', 'SE.estimate')
```

```
##        session
## model       49     50     51     52     53     54
##   fits1
##   fits2 0.4432 0.4432 0.4432 0.4432 0.4432 0.4432
##   fits3 0.4640 0.4640 0.4640 0.4640 0.4640 0.4640
```

```
AIC(spOV, spmOV)
```

```
##                                     model npar rank logLik   AIC  AICc  dAIC AICwt
## spmOV lambda0~t phi~t f~t sigma~1 move.a~t   22   22 -12305 24654 24658   0.0     1
## spOV           lambda0~t phi~t f~t sigma~1   17   17 -12399 24832 24834 177.6     0
```

## Stratified analyses

Stratification is a new feature in **openCR** 2.0 so it is treated separately here.

For a simple example we use `stratify` to split the dipper dataset into male and female strata.

```
dipperCHs <- stratify(dipperCH, covariate = 'sex')
fit <- openCR.fit(dipperCHs, model = p~stratum, stratified = TRUE)
print(make.table(fit, 'p', strata = 1:2, collapse = TRUE), na = '.')
```

```
##              session
## model.stratum 1981   1982   1983   1984   1985   1986   1987
##    fit.Male      . 0.9243 0.9243 0.9243 0.9243 0.9243 0.9243
##    fit.Female    . 0.8801 0.8801 0.8801 0.8801 0.8801 0.8801
```

As expected, these estimates are almost exactly the same as those from the earlier model `dipper.p.sex`.

`stratify` may also be used to assemble a stratified dataset from a list of capthist objects, possibly differing in detector layout. In this case it may be appropriate for the mask argument of `openCR.fit()` to be a list with a different mask for each stratum.

## References


Bishop, J. A., Cook, L. M., and Muggleton, J. (1978). The response of two species of moth to industrialization in northwest England. II. Relative fitness of morphs and population size. *Philosophical Transactions of the Royal Society of London* **B281**, 517–540.

Crosbie, S. F. (1979) *The mathematical modelling of capture–mark–recapture experiments on animal populations*. Ph.D. Thesis, University of Otago, Dunedin, New Zealand.

Crosbie, S. F. and Manly, B. F. J. (1985) Parsimonious modelling of capture–mark–recapture studies. *Biometrics* **41**, 385–398.

Ergon, T., Ergon, R., Begon, M., Telfer, S. and Lambin, X. (2011) Delayed density- dependent onset of spring reproduction in a fluctuating population of field voles. *Oikos* **120**, 934–940.

Ergon, T. and Lambin, X. (2013) Data from: Separating mortality and emigration: modelling space use, dispersal and survival with robust-design spatial capture–recapture data. Dryad Digital Repository. doi: 10.5061/dryad.r17n5

Ergon, T. and Gardner, B. (2014) Separating mortality and emigration: modelling space use, dispersal and survival with robust-design spatial capture–recapture data. *Methods in Ecology and Evolution* **5**, 1327–1336.

Laake, J.L., Johnson, D. S. and Conn, P.B. (2013) marked: An R package for maximum-likelihood and MCMC analysis of capture-recapture data. *Methods in Ecology and Evolution* **4**, 885–890.

Lebreton, J.-D., Burnham, K. P., Clobert, J., and Anderson, D. R. (1992) Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs* **62**, 67–118.

Link, W. A. and Barker, R. J. (2005) Modeling association among demographic parameters in analysis of open-population capture–recapture data. *Biometrics* **61**, 46–54.

Link, W. A. and Barker, R. J. (2010) *Bayesian Inference with Ecological Applications*. Academic Press, Amsterdam.

Marzolin, G. (1988) Polygynie du Cincle plongeur (*Cinclus cinclus*) dans les côtes de Lorraine. L'Oiseau et la Revue Francaise d'Ornithologie 58:277-286.

Nichols, J. D., Pollock, K. H. and Hines, J. E (1984) The use of a robust capture–recapture design in small mammal population studies: a field example with *Microtus pennsylvanicus*. *Acta Theriologica* **29**, 357–365.

Pledger, S., Pollock, K. H. and Norris, J. L. (2010) Open capture–recapture models with heterogeneity: II. Jolly-Seber model. *Biometrics* **66**, 883–890.

Pollock, K. H., Nichols, J. D., Brownie, C. and Hines, J. E. (1990) Statistical inference for capture–recapture experiments. *Wildlife Monographs* **107**. 97pp.

Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* **52**, 703–709.

Schwarz, C. J. and Arnason, A. N. (1996) A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics* **52**, 860–873.

Schofield, M. and Barker, R. (2016) 50-year-old curiosities: ancillarity and inference in capture–recapture models. *Statistical Science* **31**, 161–174.

Seber, G. A. F. (1982) *The estimation of animal abundance and related parameters*. 2nd edition. Griffin.

Williams, B. K., Nichols, J. D. and Conroy, M. J. (2002) *Analysis and management of animal populations*. Academic Press, San Diego.