

openCR 1.4 - open population capture–recapture

Murray Efford

2019-06-30

Contents

1	Outline	2
1.1	Model types	2
1.2	Data	3
1.3	Model specification and fitting	3
1.4	Parameterization	3
1.5	Features and limitations	3
2	Dipper example	3
3	A brief survey of open population capture–recapture models	5
3.1	CJS vs JS	5
3.2	Parameterization of recruitment in JSSA models	5
3.3	Conditional vs full likelihood JSSA	6
3.4	Sufficient statistics vs capture histories	6
3.5	Robust design	6
3.6	Spatial vs nonspatial	6
3.7	Home-range shifts between primary sessions	7
4	Data structure and input	7
5	Model types	7
5.1	Non-spatial openCR models	8
5.2	Spatial openCR models	9
6	Model formulae	10
6.1	Built-in predictors	10
6.2	User-provided covariates	11
7	More on modelling	11
7.1	Closed populations	11
7.2	Finite mixtures	12
7.3	Age	12
7.4	Sampling intervals	12
7.5	Custom design data	12
7.6	Transience	14
7.7	Factor coding	15
7.8	Mean of a parameter across levels of a factor	16
8	Movement models	16
8.1	Movement kernels	16
8.2	User-defined kernel	17
8.3	Plotting and summary	17
8.4	Warnings	18
9	Derived parameters	18

10 Simulating open-population data	20
11 Troubleshooting	20
11.1 Nonidentifiability	20
11.2 Failure of numerical maximization	21
11.3 Speed	22
12 Extras	22
12.1 Sampling variance warning	22
12.2 Example datasets	23
12.3 Testing assumptions	23
12.4 Limitations of openCR	24
12.5 Differences from secr	24
12.6 Relationship to other software	25
13 References	26

The R package **openCR** fits both non-spatial and spatial capture–recapture models to data from open animal populations, where there is turnover during sampling. The interface generally resembles that of **secr** (Efford 2018a) upon which **openCR** depends for some functions. This document explains the purpose and general features of **openCR**. Help pages should be consulted for more detail on particular functions. Worked examples using published datasets are given in another vignette `openCR-examples.pdf`, and simulation-based examples are in a further vignette `openCR-simulations.pdf`¹. A paper describing the spatial model has been submitted (Efford and Schofield 2019).

This is still something of a work in progress, so be careful to check results ‘make sense’ and be aware of limitations.

1 Outline

1.1 Model types

openCR fits nonspatial open-population models of the Cormack–Jolly–Seber (CJS) and Jolly–Seber–Schwarz–Arnason (JSSA² or ‘POPAN’) types. JSSA models are offered in both full and conditional likelihood forms, each with several parameterizations of recruitment, and incorporating Pollock’s robust design. Pradel analyses are also provided.

Spatial versions of the CJS and JSSA model types are also provided³. The spatial models allow for ‘multi’, ‘proximity’ or ‘count’ detectors as defined in **secr**. Several functions are implemented for the decline in hazard of detection with distance. Movement between primary sessions may be modelled (cf Ergon and Gardner 2014; Glennie et al. 2019), but particular care is needed, especially with respect to kernel truncation.

¹These supplementary vignettes are not included with the package. It is intended to distribute them on the website www.otago.ac.nz/density. Otherwise contact the author.

²As far as I know, this abbreviation was first used by Pledger et al. (2010). Recognising the contributions of Crosbie and Manly, Schofield and Barker (2009) and Cowen et al. (2010) referred to it the Crosbie–Manly–Arnason–Schwarz (CMAS) model. Link and Barker (2010) used ‘Crosbie–Manly–Schwarz–Arnason’ (CMSA) for the same model. CMSA has since been used by various authors, including Schofield and Barker (2016). JSSA is used in **openCR** because this highlights its evolution from the widely known Jolly–Seber model. POPAN refers to the software of Schwarz and Arnason (1996), recycled as the name of a data type in MARK.

³The utility of the spatial CJS model type (CJSsecr) is in doubt because the distribution of detected animals is not uniform at first detection, but rather biased towards the vicinity of the detectors.

1.2 Data

Data are assumed to be from a robust design. Secondary sampling sessions are nested within primary sessions and all turnover (births, deaths, immigration or emigration) is between primary sessions (Pollock 1982). There may be a single secondary session per primary session (this limits identifiability of some parameters).

1.3 Model specification and fitting

Models are specified using formula notation as in **secr**. Possible predictors include both pre-defined variables for learned responses, trend over time, etc., and user-provided covariates. Models are fitted by numerically maximizing the log likelihood. The likelihood is formed as a product over capture histories (Pledger et al. 2010) rather than from summary statistics. The fitted model is an object of class ‘openCR’ for which generic methods are implemented (`print`, `predict`, `AIC`, `plot` etc.).

Variation in a parameter between primary sessions is modelled as e.g., `model = phi ~ session`⁴. Within-session variation in detection parameters may also be modelled (see field vole example in `openCR-examples.pdf`).

1.4 Parameterization

A selection of parameterizations is offered for recruitment in JSSA models. Models can also be parameterized in terms of the time-specific population size (non-spatial models) or density (spatial models), avoiding the super-population parameter.

Super-population size (or density in the case of **secr** models) may be computed as a derived parameter from ‘CL’ models with the function `derived()`, which also computes time-specific population sizes and densities.

1.5 Features and limitations

openCR has definite limitations that may or may not be addressed in future versions. Important differences between **secr** and **openCR** are noted here. Online help is not guaranteed: users should attempt to solve their own problems, or seek help from other users via `phidot` or `secr` group.

2 Dipper example

We start with a simple nonspatial example. Lebreton et al. (1992) demonstrated Cormack-Jolly-Seber methods with a dataset on European Dipper (*Cinclus cinclus*) collected by Marzolin (1988). The object `dipperCH` distributed with **openCR** provides these data in the **secr** ‘capthist’ format. See the Examples section of its help page `?dipperCH` for code to input the data from other sources.

```
library(openCR)           # also loads secr
options(digits = 4, width = 90) # for more readable output
```

Dippers were captured annually over 1981–1987.

```
m.array(dipperCH, never.recap = T) # compare Lebreton et al. 1992 Table 10
```

```
##      R 1982 1983 1984 1985 1986 1987 NRecap
## 1981 22   11    2    0    0    0    0     9
## 1982 60           24    1    0    0    0    35
```

⁴This is equivalent of `~t` in Lebreton et al. (1992) or `~time` in **RMark**, and **openCR** recognises `~ t` as a synonym of `~ session`.

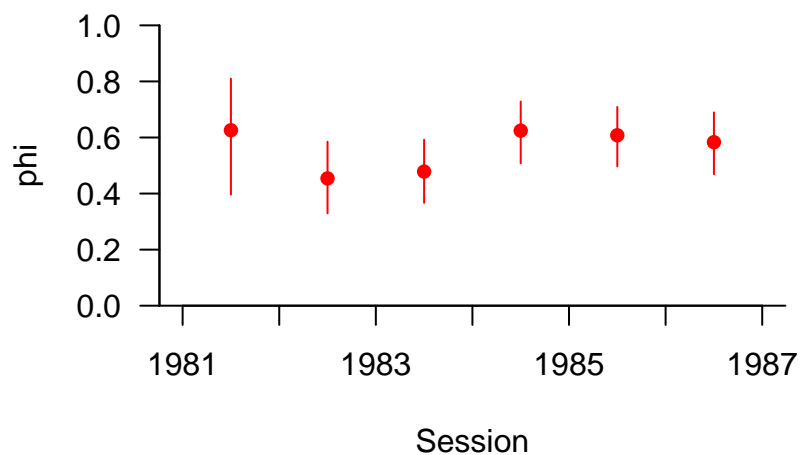
```
## 1983 78          34    2    0    0    42
## 1984 80          45    1    2    32
## 1985 88          51    0    37
## 1986 98          52    46
## 1987 93          93
```

We can fit a Cormack-Jolly-Seber model directly with `openCR.fit` and display the estimates:

```
dipper.phi.t <- openCR.fit(dipperCH, type = 'CJS', model = phi~t)
predict(dipper.phi.t)
```

```
## $p
##      session estimate SE.estimate    lcl    ucl
## 1981      1      NA          NA      NA      NA
## 1982      2  0.9021    0.02906  0.8286  0.9461
## 1983      3  0.9021    0.02906  0.8286  0.9461
## 1984      4  0.9021    0.02906  0.8286  0.9461
## 1985      5  0.9021    0.02906  0.8286  0.9461
## 1986      6  0.9021    0.02906  0.8286  0.9461
## 1987      7  0.9021    0.02906  0.8286  0.9461
##
## $phi
##      session estimate SE.estimate    lcl    ucl
## 1981      1  0.6258    0.11165  0.3965  0.8098
## 1982      2  0.4542    0.06662  0.3295  0.5849
## 1983      3  0.4784    0.05845  0.3669  0.5921
## 1984      4  0.6244    0.05703  0.5079  0.7281
## 1985      5  0.6079    0.05483  0.4970  0.7088
## 1986      6  0.5833    0.05721  0.4688  0.6895
## 1987      7      NA          NA      NA      NA
```

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



From this example you can see some of the virtues of **openCR**

- accessible data summaries

- compact model specification
- direct plotting and tabulation of results.

See `openCR-examples.pdf` for more extensive analyses of this dataset.

3 A brief survey of open population capture–recapture models

There is a large literature on open-population capture–recapture modelling. Almost all modern models derive from the Cormack–Jolly–Seber (CJS) or Jolly–Seber (JS) models (Seber 1982), with refinements by Crosbie and Manly (1985), Schwarz and Arnason (1996), Pradel (1996) and others. The MARK software (White and Burnham 1999) implemented many of these developments and remains the standard. This section describes differences among models as they relate to **openCR**.

3.1 CJS vs JS

The split between the CJS and JS model lineages is fundamental. CJS models do not model the first capture of each animal; they condition on that capture and model subsequent recapture probabilities p and apparent survival ϕ . CJS estimates of apparent survival are robust and useful (Lebreton et al. 1992), but CJS models stop short of estimating abundance, recruitment or population trend.

JS models model the first capture of each animal, and lead either directly or indirectly to estimates of abundance and recruitment. The modern development of JS methods rests heavily on Schwarz and Arnason (1996), so **openCR** follows Pledger et al. (2010) in using the label ‘JSSA’. JSSA models were the basis of the POPAN software, which led to the POPAN data type in MARK. JSSA models are the main focus of **openCR**.

3.2 Parameterization of recruitment in JSSA models

The JSSA model appears in several different forms whose unity is obscured by differing parameterizations of recruitment. The classic POPAN formulation uses entry probabilities: the members of a notional superpopulation enter the population with time-specific probability β_j (PENT in MARK), an idea from Crosbie and Manly (1985). Other parameterizations are

- number of new entrants at each time j
- per capita fecundity (new entrants at time j scaled by $1/\text{number in population at } j - 1$)
- seniority (reverse-time survival Pradel 1996, Nichols 2016)
- population growth rate λ
- (relative) number in population at each time j

Estimates of recruitment or implied recruitment from any one of these six parameterizations can be used to infer the others⁵. The choice of parameterization rests on which is more natural for the problem in hand (and allows the desired constraints to be applied) and on practicalities (some are more likely to give numerical problems than others).

Schwarz (2001) is illuminating (see also chapter on Jolly–Seber models by Schwarz and Arnason in the MARK book, Cooch and White 2017). Pradel (1996), Williams, Nichols and Conroy (2002: p.518 et seq.), Pledger et al. (2003, 2010) and Link and Barker (2005) also comment on and compare JS parameterizations. See also the MARK help page on ‘Recruitment Parameters in Jolly–Seber models’ (‘Recruitment Parameters’ in the help index).

⁵except for some mostly trivial differences relating to removals

3.3 Conditional vs full likelihood JSSA

For each JSSA recruitment parameterization there is a choice between models that include the total number of detected individuals (u . or n in different notations), and models that condition on this number. Conditional-likelihood models do not directly estimate abundance; abundance is estimated as a derived parameter (Schwarz and Arnason 1996). Full-likelihood models include abundance as a parameter. The choice of formulation has virtually no effect on the parameter estimates⁶. The conditional likelihood form is somewhat faster and easier to fit (Schwarz and Arnason 1996), and it focuses on parameters that are estimated robustly (apparent survival, seniority, population growth rate).

The conditional models discussed by Pradel (1996), Link and Barker (2005), Schofield and Barker (2016) and others lack a distinguishing label to indicate their collective similarity. The label Pradel–Link–Barker PLB is suggested by Efford and Schofield (in revision).

3.4 Sufficient statistics vs capture histories

Historically the CJS and JS likelihoods have been expressed in terms of ‘sufficient statistics’ that are time-specific counts of animals in different categories, such as the number caught, the number marked etc. This approach is used in the **openCR** function `JS.direct` and with the Pradel model type in `openCR.fit`. The likelihood may also be computed as a product over terms, one for each observed capture history⁷. Modelling of individual capture histories, is slower, but it is extremely flexible, allowing direct inclusion of censoring, learned responses, individual covariates, secondary sessions and other extensions. This is the approach used in MARK and `openCR.fit`.

3.5 Robust design

Most published formulations of CJS and JSSA models admit only one secondary session per primary session. Data collected according to a robust design with multiple secondary sessions must be collapsed to a single sample per primary session. However, it is simple to adapt the capture-history models for multiple secondary occasions, and this makes better use of the data. MARK offers many specific robust design models. A robust design is assumed in **openCR**; data with a single secondary session per primary session are merely a special case.

3.6 Spatial vs nonspatial

Models may be spatially explicit or not. Nonspatial models ignore the spatial distribution of animals. Spatial models use the spatially explicit capture–recapture paradigm of Efford (2004), Borchers and Efford (2008) and Royle et al. (2014). Open population spatial models using MCMC were published by Gardner et al. (2010), Chandler and Clark (2014), Ergon and Gardner (2014), Whittington and Sawaya (2015) and others. Glennie et al. (2019) proposed a frequentist hidden Markov formulation. The spatial models in **openCR** are described by Efford and Schofield (in revision) and provide very similar estimates to those of Glennie et al. (2019).

There are three major motivations for open spatial models

- allowance for varying extent of sampling area
- modelling of individual heterogeneity due to differential access to detectors
- separation of emigration and mortality

openCR fits spatial analogues of CJS and JSSA models by maximizing the likelihood. The abundance parameter is density D (animals per hectare) rather than population size N .

⁶this may not be true for spatial models with spatially varying density, but these models are not considered in **openCR**.

⁷strictly, the product over observed histories is only one component of the likelihood

Recruitment in spatial models may be modelled using parameterizations to those described above for non-spatial models, replacing ‘number’ by ‘density’. The locations at which animals recruit are not modelled.

3.7 Home-range shifts between primary sessions

By definition, the interval between primary sessions is long enough for turnover due to births and deaths. It is also possible that resident animals shift their home ranges (i.e. disperse). Spatial models may either ignore such movement (Gardner et al. 2010, Chandler and Clark 2014, Whittington and Sawaya 2015) or attempt to model it (Ergon and Gardner 2014). There are good arguments for modelling movement:

- Movement that is ignored inflates estimates of the within-session scale of detection σ , with flow-on effects on demographic parameters.
- If the distribution of dispersal distances can be inferred from the detection histories of residents then it is possible in principle to separate actual mortality from losses due emigration (Ergon and Gardner 2014). However, the robustness and data requirements of movement models have yet to be fully understood.

4 Data structure and input

Data should be provided to `openCR.fit` as `secr` ‘capthist’ objects. The occasions of a single-session dataset are treated as open-population temporal samples. For spatial analyses, the capthist object should use a point detector type (‘multi’, ‘proximity’ or ‘count’).

`openCR` mostly uses the terminology of primary and secondary sessions (Pollock 1982) rather than ‘session’ and ‘occasions’ as in `secr`. Where ‘session’ appears without qualifier it refers to a primary session composed of one or more secondary sessions.

The optional `intervals` attribute of the capthist object defines the structure. If intervals are not specified then they default to 1.0 and each occasion is treated as a primary session. If intervals are specified then some may be zero; occasions separated by ‘zero’ intervals are treated as secondary sessions within the same primary session, as in MARK.

A multi-session capthist object will be converted automatically to a single-session object using function `secr::join`. An appropriate intervals attribute is constructed, using the intervals attribute of the multi-session object for the intervals between primary sessions (1.0 if not specified), and setting other intervals to zero.

To construct your own capthist objects –

1. Consult `secr-datainput.pdf`, or
2. Convert a dataframe in RMark input format using `secr::unRMarkInput`, or
3. Read a MARK `.inp` input file with `read.inp`.

Examples of data input code also appear on the help pages for data objects `FebpossumCH`, `fieldvoleCH`, `microtusCH` and `dipperCH`.

5 Model types

The various models available in `openCR` are named to encode the distinctions made in the ‘Brief survey’. Names are formed by concatenating four components:

1. ‘CJS’ vs ‘JSSA’
2. Spatial (‘secr’) vs non-spatial (default, blank)
3. JSSA recruitment parameterization (‘f’, ‘l’, ‘b’, ‘g’, ‘BN’, ‘BD’, ‘N’, ‘D’ - see following)
4. JSSA likelihood conditional (‘CL’) vs full (default, blank)

Thus ‘JSSAsecrfCL’ is a spatial JSSA model parameterized in terms of per capita recruitment f and fitted by maximizing the conditional likelihood (a spatial version of Link and Barker (2005), minus parameter covariation). Any movement model is specified separately with the ‘movementmodel’ argument of `openCR.fit`.

Parameters vary with the type of model, as listed below. Each of these primary parameters (‘real’ parameters in MARK) may also be modelled as a linear combination of predictors on a suitable link scale, allowing the inclusion of covariates and constraints. The coefficients of the parameter-specific linear combinations are called ‘beta’ parameters in MARK; the likelihood is maximized with respect to the concatenated list of beta parameters.

5.1 Non-spatial openCR models

5.1.1 Parameters and model types

Table 1. Parameter definitions and default link functions (nonspatial models)

Parameter	Symbol	Link	Description
p	p	logit	capture probability (recapture probability for CJS)
phi*	ϕ	logit	apparent survival
b	b	mlogit	entry probability of PENT in MARK
f*	f	log	per capita recruitment rate
gamma*	γ	logit	seniority (Pradel 1996)
lambda*	λ	log	population growth rate (finite rate of increase)
superN	N	log	superpopulation size
BN	B_N	log	number of entrants
N	N_j	log	time-specific population size

* parameters marked with an asterisk are scaled by the interval between primary sessions.

Table 2. Parameters of nonspatial **openCR** models

Type	p	phi	b	f	gamma	lambda	superN	BN	N
CJS	+	+							
JSSAbCL	+	+	+						
JSSAfCL	+	+		+					
JSSAgCL	+	+			+				
JSSAICL	+	+				+			
JSSAb	+	+	+				+		
JSSAf	+	+		+			+		
JSSAg	+	+			+		+		
JSSAI	+	+				+	+		
JSSAB	+	+						+	
JSSAN	+	+							+

Models with type ending in CL are of the Pradel–Link–Barker type.

5.1.2 Non-spatial models using sufficient statistics

openCR mostly fits models by modelling capture histories one-by-one. An alternative faster method is to evaluate the likelihood expressed in terms of sufficient statistics. Sufficient statistics vary among models, but

they are typically counts such as provided by the function `JS.counts`. The ‘sufficient statistics’ approach is not compatible with individual covariates. The non-spatial model types ‘Pradel’ and ‘Pradelg’ are implemented in **openCR** using sufficient statistics (Pradel 1996) and therefore fall outside the main framework (Table 2). They correspond to ‘JSSAICL’ and ‘JSSAgCL’ respectively, and estimate the same parameters as those models. Estimates should coincide except when there are losses on capture. ‘Pradel’ is parameterized in terms of population growth rate (lambda) and ‘Pradelg’ is parameterized in terms of seniority (gamma).

Additionally, the function `JS.direct` computes classic Jolly–Seber estimates using the sufficient statistics.

5.2 Spatial openCR models

Table 3. Parameter definitions and default link functions (spatial models)

Parameter	Symbol	Link	Description
lambda0	λ_0	log	detection function intercept
sigma	σ	log	detection function scale (m)
z	z	log	detection function shape parameter (HHR, HAN, HCG, HVP)
phi*	ϕ	logit	apparent survival
b	b	mlogit	entry probability (beta)
f*	f	log	per capita recruitment rate
gamma*	γ	logit	seniority (Pradel 1996)
lambda*	λ	log	population growth rate (finite rate of increase)
superD	D	log	superpopulation density
BD	B_D	log	entrants per hectare
D	D_j	log	time-specific population density

* parameters marked with an asterisk are scaled by the interval between primary sessions.

Table 4. Parameters of spatial **openCR** models

Type	lambda0	sigma	z	phi	b	f	gamma	lambda	superD	BD	D
CJSsecr	+	+	+	+							
JSSAsecrbCL	+	+	+	+	+						
JSSAsecrfCL	+	+	+	+		+					
JSSAsecrgCL	+	+	+	+			+				
JSSAsecr1CL	+	+	+	+				+			
JSSAsecrb	+	+	+	+	+				+		
JSSAsecrf	+	+	+	+		+			+		
JSSAsecrg	+	+	+	+			+		+		
JSSAsecr1	+	+	+	+				+	+		
JSSAsecrB	+	+	+	+						+	
JSSAsecrD	+	+	+	+							+
secrCL	+	+	+								
secrD	+	+	+						+		

Spatial models with type ending in CL have features in common with the Pradel–Link–Barker models, hence the label ‘spatial PLB’.

6 Model formulae

Formulae define a linear model for each ‘real’ parameter (p, phi, sigma etc.) on the link scale (logit, log etc.). Alternative link functions not shown in Tables 1 and 3 are ‘loglog’ and ‘sin’, both as defined in MARK.

The default linear combination for each parameter is a constant, null model (~1, parameter constant over time, unaffected by individual differences etc.). To include other effects build formulae using either predefined (built-in) predictors listed here, or the names of covariates.

6.1 Built-in predictors

Table 5. Built-in predictors (‘sessions’ refers to primary sessions)

Predictor	Parameters	Description
session	all except ‘superN’, ‘superD’	Factor, one level per primary session
t	all except ‘superN’, ‘superD’	synonym of ‘session’
Session	all except ‘superN’, ‘superD’	Continuous time
b	p, phi, lambda0, sigma	learned response (persists across sessions)
B	p, lambda0, sigma	detector-specific transient (Markovian) response across sessions
bk	p, phi, lambda0, sigma	detector-specific learned response (persists across sessions)
bsession	p, lambda0, sigma	learned response within sessions
Bsession	p, lambda0, sigma	transient (Markovian) response within sessions
bksession	p, lambda0, sigma	detector-specific learned response within sessions
Bksession	p, lambda0, sigma	detector-specific transient (Markovian) response within sessions
h2	all except abundance	2-class finite mixture
h3	all except abundance	3-class finite mixture
age	all except abundance	age factor
Age	all except abundance	linear effect on age
Age2	all except abundance	linear effect on age ²

Differences among the various learned responses may be understood by examining their effect on the parameter index array (PIA). This table illustrates the PIA slice corresponding to an individual with the non-spatial detection history shown (4 primary sessions, each of 4 secondary sessions). The values ‘1’ and ‘2’ refer to different parameter combinations, most commonly to levels of lambda0.

Detection history : 0100 0000 0000 0100		
~bsession	1122 1111 1111 1122	persistent within primary session
~Bsession	1121 1111 1111 1121	transient within primary session
~b	1122 2222 2222 2222	persistent
~B	1122 2222 1111 1122	transient across primary sessions

IMPORTANT NOTE: Learned response predictors (‘b’, ‘bsession’ etc.) were re-defined in **openCR** 1.3.0. Models fitted with earlier versions should be re-fitted.

6.2 User-provided covariates

The rules for covariates largely follow **secr** ([secr-overview.pdf](#)). Covariates may be at the level of primary session, secondary session (detection parameters only), individual (CL models only), or detector (spatial models only). Further complexity may be modelled by providing custom design data cutting across these categories (see below).

Individual and detector covariates are named columns in the ‘covariates’ attributes of the respective `capthist` and `traps` object. Covariate names should differ from the built-in predictors (Table 5).

Primary session covariates are provided to `openCR.fit` in the argument ‘`sessioncov`’, rather than associated with a data object. If ‘`sessioncov`’ is a vector (length equal to number of primary sessions) rather than a dataframe then it may be referenced as ‘`scov`’ in model formulae.

Covariates for detection parameters in secondary sessions are provided in the ‘`timecov`’ argument; if ‘`timecov`’ is a vector (length equal to total number of secondary sessions) rather than a dataframe then it may be referenced as ‘`tcov`’ in model formulae.

7 More on modelling

7.1 Closed populations

The types ‘`secrD`’ and ‘`secrCL`’ cause `openCR.fit` to treat the data as if from a closed population (no mortality, no recruitment, no movement); the `intervals` attribute is ignored. **openCR** 1.4 fits simple closed models faster than **secr** 3.2. This surprising result is due to multithreading, improved coding, and avoidance of some modelling complexity.

```
msk <- make.mask(traps(captdata), buffer = 100, type = 'trapbuffer')

secr <- secr.fit(captdata, detectfn = 'HHN', mask = msk, trace = FALSE)
openCR <- openCR.fit(captdata, detectfn = 'HHN', mask = msk, type = 'secrD')

# massage the predict.openCR results to the same format as predict.secr
pred_openCR <- plyr::rbind.fill(predict(openCR))[c(2,1,3),-1]
rownames(pred_openCR) <- secr$realnames

# compare estimates
predict(secr)[-1]

##          estimate SE.estimate    lc1    uc1
## D             5.485    0.64703  4.356  6.9058
## lambda0       0.307    0.03413  0.247  0.3815
## sigma        28.764    1.30055 26.326 31.4283

pred_openCR

##          estimate SE.estimate    lc1    uc1
## D             5.485    0.64479  4.356  6.9058
## lambda0       0.307    0.03403  0.247  0.3815
## sigma        28.764    1.29988 26.326 31.4283

# compare timings in seconds
c(secr = secr$proctime, openCR = openCR$proctime)

##   secr.elapsed openCR.elapsed
##             18.37           4.99
```

The maximised log likelihoods differ because **openCR** does not include the multinomial constant. **secr** has function `logmultinom` that lets us add it back:

```
# compare maximised log likelihoods
c(secr.logLik = logLik(secr), openCR.logLik = logLik(openCR) + logmultinom(captdata))
```

```
##   secr.logLik openCR.logLik
##      -758.9      -758.9
```

7.2 Finite mixtures

Two- and three-class finite mixtures (h2, h3) allow for individual heterogeneity in detection and turnover parameters (Pledger et al. 2003, 2010). Using one of these predictors in a formula causes a further real parameter ‘pmix’ to be added. pmix is the proportion in latent mixture class 2 for h2, and the proportions in classes 2 and 3 for h3 (the proportion in class 1 is obtained by subtracting from 1). The implementation in **openCR** assumes that class membership applies across all parameters. The posterior probabilities of class membership for all detected individuals are returned as the ‘posterior’ component of the fitted model.

Finite mixture likelihoods are prone to multimodality. Misleading estimates result when the numerical maximization settles on a local maximum (see also [secr-finitemixtures.pdf]).

7.3 Age

If age is modelled as a factor then it is useful to group older animals in a maximum age class (‘maximumage’). Animals are assumed to be first caught at age zero unless ‘initialage’ is specified. ‘maximumage’ and ‘initialage’ are optional components of the ‘details’ argument of ‘openCR.fit’. ‘initialage’ can name an individual covariate to avoid the assumption that all animals are the same age at first detection.

For a quadratic relationship with age, specify an additive model with both Age and Age2 terms (e.g., model = phi ~ Age + Age2).

7.4 Sampling intervals

We have seen the role of the intervals attribute in defining primary and secondary sessions. Between-session intervals need to be specified only if they vary, or if you would like rates (phi, gamma, lambda, f) to be reported in time units other than the (implicitly constant) sampling interval. Scaling from the standardised parameter θ_j to the interval-specific value θ'_j uses $\theta'_j = \theta_j^{T_j}$ where θ_j is one of ϕ_j or λ_j , and T_j is the duration of interval j .

Scaling γ follows the same pattern except that the relevant duration for γ_j is T_{j-1} . Scaling per capita recruitment f_j is more tricky. We use $f'_j = (\phi_j + f_j)^{T_j} - \phi_j^{T_j}$.

7.5 Custom design data

Occasionally there is a need for covariates that do not relate specifically to individuals, sessions or detectors, and are not included as canned predictors. For this you must construct your own dataframe of design data and pass it as the ‘dframe’ argument of `openCR.fit`. Design data are used as input to the `model.matrix` function (the ‘data’ argument); `model.matrix` generates the design matrix for each real parameter. Design data are usually constructed internally in `openCR.fit` from named covariates and other predictors that appear in model formulae; if ‘dframe’ is provided then the internally constructed design data are added as extra columns, overwriting any custom columns of the same name. The same design dataframe is used for all parameters.

Constructing ‘dframe’ is fiddly. The dataframe should have one row for each combination of unique capture history, secondary session, detector and latent class (mixture). For nonspatial models without finite mixtures this collapses to one row for each capture history and secondary session. The order of rows follows that of the elements in an array with dimensions (n, S, K, X) for n unique capture histories, S secondary sessions, K detectors and X latent classes⁸. The `secr` function `insertdim` can help to expand data into the correct row order.

A warning: by default `openCR.fit` replaces the input `capthist` with a more compact version using only unique capture histories (the number of each is kept in the individual covariate ‘freq’; see the function `squeeze`). Design data are in terms of the ‘squeezed’ capture histories.

In this example we define a function to construct custom design data for a learned response.

```
makedf.b <- function (ch, spatial = FALSE, nmix = 1, naive = FALSE) {
  ch <- squeeze(ch)
  # Construct matrix of logical values TRUE iff caught before
  detected <- apply(abs(ch), 1:2, sum) > 0
  detected <- t(apply(detected, 1, cumsum) > 0)
  if (naive)
    b <- rep(FALSE, prod(dim(ch)[1:2]))
  else
    b <- t(apply(detected, 1, function(x) {x[which.max(x)] <- FALSE; x}))
  # For a simple non-spatial case: data.frame(customb = as.vector(b))
  # More generally:
  n <- nrow(ch)
  S <- ncol(ch)
  K <- if (spatial) dim(ch)[3] else 1
  data.frame(customb = insertdim(b, 1:2, c(n,S,K,nmix)))
}
```

Now compare the result with the canned predictor ‘b’ for a persistent learned response.

```
ovenj <- join(ovenCH)
fitb <- openCR.fit(ovenj, model = p ~ b)
fitbc <- openCR.fit(ovenj, model = p ~ customb, dframe = makedf.b(ovenj))
AIC(fitb, fitbc)
```

```
##           model npar rank logLik   AIC  AICc  dAIC  AICwt
## fitb      p~b phi~1   3    2 -254.6 515.2 515.6    0    0.5
## fitbc    p~customb phi~1   3    2 -254.6 515.2 515.6    0    0.5
```

Our custom model gives exactly the same result as the canned predictor ‘b’ when `type = ‘CJS’` because the precise secondary session of first capture is irrelevant for CJS models (recaptures are modelled only for subsequent primary sessions unless `details$CJSp1 == TRUE`).

Discrepancies can arise with non-CJS models because these account for animals never detected. The corresponding likelihood component uses a distinct design matrix for a ‘naive’ animal. To customize non-CJS models a separate `dframe` should be provided that applies to naive animals:

```
fitb2 <- openCR.fit(ovenj, model = p ~ b, type = 'JSSAfCL', start = fitb)
fitbc2 <- openCR.fit(ovenj, model = p ~ customb, type = 'JSSAfCL',
  dframe = makedf.b(ovenj), dframe0 = makedf.b(ovenj, naive = TRUE))
AIC(fitb2, fitbc2)
```

```
##           model npar rank logLik   AIC  AICc  dAIC  AICwt
```

⁸This rectangular (or cuboidal) configuration includes cells that are redundant and unused for a particular model type (e.g., cells corresponding to sessions at or before first capture in CJS models). However, the full complement of rows is required in `dframe`.

```
## fitbc2 p~customb phi~1 f~1 4 1 -661.2 1330 1331 0.00 1
## fitb2 p~b phi~1 f~1 4 4 -670.9 1350 1350 19.43 0
```

7.6 Transience

An ad hoc adjustment for transience may be programmed as follows (cf Pradel et al. 1997).

```
makedf.resident <- function (ch, spatial = FALSE, nmix = 1) {
  ch <- squeeze(ch)
  n <- nrow(ch)
  S <- ncol(ch)
  K <- if (spatial) dim(ch)[3] else 1
  primary <- primarysessions(intervals(ch))
  detected <- apply(abs(ch), 1:2, sum)>0
  nprimary <- apply(detected, 1, function(x) length(unique(primary[x])))
  data.frame(resident = insertdim(nprimary>1, 1, c(n,S,K,nmix)))
}
```

A simpler approach is to code an individual covariate that scores whether an individual was detected in more than one primary session.

```
addressidentcov <- function (ch) {
  primary <- primarysessions(intervals(ch))
  detected <- apply(abs(ch), 1:2, sum)>0
  nprimary <- apply(detected, 1, function(x) length(unique(primary[x])))
  covariates(ch) <- data.frame(residentcov = nprimary>1)
  ch
}
```

Results are identical:

```
ovenj <- join(ovenCH)
ovenj <- addressidentcov(ovenj)
fitnull <- openCR.fit(ovenj, model = phi ~ 1)
fitcov <- openCR.fit(ovenj, model = phi ~ residentcov)
fitdf <- openCR.fit(ovenj, model = phi ~ resident, dframe = makedf.resident(ovenj))
fits <- openCRlist(fitnull, fitcov, fitdf)
AIC(fits)
```

```
##          model npar rank logLik  AIC  AICc  dAIC  AICwt
## fitcov  p~1 phi~residentcov  3   2 -225.8 457.6 458.0  0.00  0.5
## fitdf   p~1 phi~resident    3   2 -225.8 457.6 458.0  0.00  0.5
## fitnull  p~1 phi~1         2   2 -254.6 513.2 513.4 55.56  0.0
```

```
pred <- predict(fits, newdata = data.frame(resident = TRUE, residentcov = TRUE))
do.call(rbind, lapply(pred, '[[' , 'phi'))
```

```
##          resident residentcov estimate SE.estimate  lcl  ucl
## fitnull    TRUE          TRUE  0.4630    0.05473 0.3590 0.5703
## fitcov     TRUE          TRUE  0.7387    0.06112 0.6031 0.8402
## fitdf     TRUE          TRUE  0.7387    0.06112 0.6031 0.8402
```

Hines et al. (2003) suggested extending the definition of residence to include animals captured at least d days apart within a primary session; either of the approaches here may be modified accordingly. Here is the code for two individual covariates:

```

addressidentcov2 <- function (ch, d = 1) {
  primary <- primarysessions(intervals(ch))
  secondary <- secondarysessions(intervals(ch))
  detected <- apply(abs(ch), 1:2, sum)>0
  nprimary <- apply(detected, 1, function(x) length(unique(primary[x])))
  dsecondary <- apply(detected, 1, function(x)
    max(by(secondary[x], primary[x], function(y) diff(range(y)))))
  covariates(ch) <- data.frame(residentcov1 = nprimary>1,
                             residentcov2 = nprimary>1 | dsecondary>=d)
  ch
}

```

7.7 Factor coding

Factor predictors take a number of discrete values (levels). These are usually represented by columns of 0's and 1's in the design matrix, where the number of columns (and coefficients) relates to the number of levels. The default in R is to use 'treatment contrasts'; one coefficient describes a reference class (level) and other coefficients represent the effect size (difference from the reference class on the link scale). By default the first level is used as the reference: for time effects (t, session) the first primary session is the reference level⁹.

This may lead to trouble if the parameter is not identifiable in the reference class. One workaround is to specify a session covariate with differently ordered levels. Another is to switch from treatment contrasts to dummy variable coding in which each coefficient represents the magnitude of one real parameter on the link scale (useful in itself). Dummy variable coding is achieved by overriding the default contrasts and removing the intercept from the formula (-1). The following model fits yield the same estimates of 'real' parameters and the same log-likelihood, but with different 'beta' parameters:

```

fit0 <- openCR.fit(ovenCH, model = p~t)
contr.none <- function(n) contrasts(factor(1:n), contrasts = FALSE)
fitd <- openCR.fit(ovenCH, model = p ~ -1+t,
                 details = list(contrasts = list(t = contr.none)))
coef(fit0)

```

```

##          beta SE.beta    lcl    ucl
## p      -1.54954  0.2459 -2.0316 -1.0675
## p.t3    0.32964  0.3280 -0.3133  0.9725
## p.t4   -1.42725  0.5259 -2.4581 -0.3964
## p.t5   -0.14372  0.4489 -1.0236  0.7361
## phi   -0.03141  0.2399 -0.5016  0.4388

```

```
coef(fitd)
```

```

##          beta SE.beta    lcl    ucl
## p.t2   -1.54955  0.2459 -2.0316 -1.0675
## p.t3   -1.21990  0.2188 -1.6487 -0.7911
## p.t4   -2.97677  0.4663 -3.8907 -2.0628
## p.t5   -1.69324  0.3783 -2.4347 -0.9518
## phi   -0.03143  0.2399 -0.5016  0.4387

```

⁹This does not apply for times when a parameter can never be estimated – for example, **openCR** understands that seniority (gamma) is not estimated for the first session and uses the second session for the reference level.

7.8 Mean of a parameter across levels of a factor

Suppose you wish to estimate the average of a parameter across levels of a factor such as time (session). Cooch and White (2017 Section 6.15) advocate modifying the design matrix so that one beta parameter (coefficient) relates directly to the mean. This is achieved very simply in `openCR.fit`¹⁰ by setting the contrast function for the factor to `contr.sum` in the `details` argument¹¹. With the resulting factor coding the first coefficient corresponds to the mean. Applying this to estimate the average time-specific survival rate for the dippers assuming constant recapture probability:

```
fit <- openCR.fit(dipperCH, model = phi~t, details = list(contrasts = list(t = contr.sum)))
invlogit(coef(fit)['phi',c('beta','lcl','ucl')])
```

```
##      beta  lcl  ucl
## phi 0.5633 0.505 0.6199
```

The mean is backtransformed from the link scale. This results in some bias owing to the nonlinearity of link functions other than the identity function. Cooch and White take the position that the bias is often ignorable.

8 Movement models

Potential movement of home ranges between primary sessions (= dispersal) is a critical part of open-population models. The argument `movementmodel` of `openCR.fit` allows the possibilities in Table 6. Two of these do not model movement at all. The default ‘static’ is a null model in which each animal retains the same home range. The ‘uncorrelated’ option models the locations of an animal independently in each primary session; information is sacrificed and no particular movement model is implied.

The remaining options (normal, exponential, t2D, uniform, and user-supplied function) fit a dispersal kernel (Nathan et al. 2012) to represent movement between primary sessions. This usually requires at least one more parameter to represent the spatial scale of dispersal.

IMPORTANT NOTE: Movement models were recoded in **openCR** 1.4.0 and results may differ from previous releases.

Table 6. Models for movement between primary sessions.

Movement model	Parameter(s)	Description
<code>static</code>	(none)	Centres constant across primary sessions
<code>uncorrelated</code>	(none)	Centres unconstrained
<code>normal</code>	<code>move.a</code>	Gaussian (bivariate normal) kernel
<code>exponential</code>	<code>move.a</code>	Laplace (negative exponential) kernel
<code>t2D</code>	<code>move.a</code> , <code>move.b</code>	2-dimensional <i>t</i> -distribution kernel
<code>uniform</code>	(none)	Uniform within kernel radius
(user function)	<code>move.a</code> , <code>move.b</code>	User-supplied kernel function (<code>ncores = 1</code> only)

8.1 Movement kernels

All movement kernels are radially symmetrical. Relative probability of movement is specified in terms of radial distance r from the point of origin (Table 7).

¹⁰This also works in `secr.fit`.

¹¹Helmert contrasts (`contr.helmert`) also yield the mean as the first coefficient, but the coding is more obscure.

The extent of the kernel is controlled by details argument ‘kernelradius’ that gives the radius in terms of mask cells. The default radius (10) results in a discretized kernel of 349 cells (square of 441 cells minus corners). Cell-specific values are normalised so that they sum to 1.0 across the kernel. Dispersal probability effectively falls to zero at the boundary of the kernel, so the kernel radius is a critical part of the model. The ‘uniform’ kernel has no parameters but depends critically on the user-specified kernel radius.

Table 7. Kernel probability density functions. From Nathan et al. (2012, Table 15.1) and Clark et al. (1999) with adjustment for parameterisation in **openCR**.

Kernel	move.a	move.b	pdf	Expected movement*	Proportion* $r > R$
normal	α_g		$\frac{1}{2\pi\alpha_g^2} \exp\left(-\frac{r^2}{2\alpha_g^2}\right)$	$\frac{\alpha_g\sqrt{\pi}}{\sqrt{2}}$	$\exp\left(-\frac{R^2}{2\alpha_g^2}\right)$
exponential	α_l		$\frac{1}{2\pi\alpha_l^2} \exp\left(-\frac{r}{\alpha_l}\right)$	$2\alpha_l$	$\exp\left(-\frac{R}{\alpha_l}\right)$
t2D	α_t	β	$\frac{\beta}{\pi\alpha_t^2} \left(1 + \frac{r^2}{\alpha_t^2}\right)^{-(\beta+1)}$	$\alpha_t \frac{\sqrt{\pi}}{2} \frac{\Gamma(\beta-0.5)}{\Gamma(\beta)}, \beta > 0.5$	$\left(\frac{\alpha_t^2}{\alpha_t^2 + R^2}\right)^\beta$

* Continuous, untruncated, kernel. Expected values for the discretized and truncated kernel will be less (see `summary.kernel`).

The ‘t2D’ kernel is the same as ‘2Dt’ of Clark et al. (1999) and Nathan et al. (2012), renamed to avoid initial digit. The parameter α_t (`move.a`) corresponds to a in Nathan et al. (2012) and \sqrt{u} in Clark et al. (1999); the parameter β (`move.b`) corresponds to $b - 1$ in Nathan et al. (2012) and p in Clark et al. (1999). Defining `move.b` as $\beta \equiv b - 1$ is handy because the default link for `move.b` (`log`) then ensures $b > 1$.

The ‘t2D’ kernel approaches bivariate normal as $\beta \rightarrow \infty$ and Cauchy as $\beta \rightarrow 0$ (e.g., Clark et al. 1999). Clark et al. (1999 p. 1485) found it hard to fit this kernel to seed dispersal data.

8.2 User-defined kernel

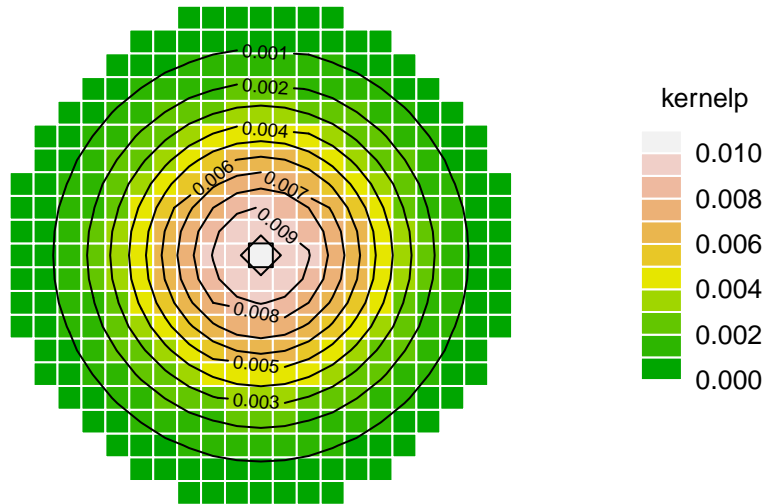
A kernel function may be specified by the user and passed in the argument `movementmodel`. The function should have argument r , and optionally a , or a and b (the last two correspond to **openCR** parameters `move.a` and `move.b`) It should return a vector of values one for each element of r , although `length(r) = 1` when the likelihood is evaluated in C++ (`details$R = FALSE`, the default). The code should give a valid result when $r = 0$. With the default link (`‘log’` for both `move.a` and `move.b`) there is no risk of $a \leq 0$ or $b \leq 0$.

8.3 Plotting and summary

A kernel may be constructed with `make.kernel` and visualised with the `plot` method. Use the `summary` method to obtain a terse description.

```
par(mar=c(2,2,3,6), cex=0.9)
k <- make.kernel(movementmodel = 'normal', spacing = 10, move.a = 40, clip = TRUE)
plot(k, contour = TRUE)
```

kernel = normal spacing = 10 kernelradius = 10 move.a = 40



```
summary(k)
```

```
## Kernel radius (cells)      : 10
## Spacing (side of cell)    : 10 (m)
## Number of cells          : 349
## Movement model           : normal
## Parameter(s)             : move.a = 40
## Proportion truncated      : 0.03189
## Expected movement (trunc) : 47.91 (m)
## Expected movement (full)  : 50.13 (m)
```

Use the `secr` function `spotHeight(k)` to display cell values on the plot.

8.4 Warnings

1. The ‘uncorrelated’ option is not recommended. It discards information on the continuity of home ranges between primary sessions, and estimates may vary with the (often arbitrary) extent of the habitat mask.
2. Kernel-based movement models require extreme care. Definitive advice cannot yet be given on the safe use of these models. Long-distance movements will usually be poorly sampled and poorly modelled.
3. User-defined functions cannot be used with multithreaded C++, so they will be slow to fit; always set `ncores = 1`.

9 Derived parameters

Various derived parameters may be computed from a fitted model. Specifically,

1. Abundance at each primary session (population size for non-spatial models or density for spatial models) may be computed from any JSSA model, including those fitted by maximizing the conditional likelihood. By default, the estimator is Horvitz-Thompson-like at the level of the superpopulation (N or D). For non-spatial models $\hat{N} = \sum_{i=1}^n \hat{p}_i^{-1}$ where \hat{p}_i is the estimated probability animal i is seen in at least one

session. For spatial models $\hat{D} = \sum_{i=1}^n \hat{a}_i^{-1}$, where \hat{a}_i is the estimated effective sampling area of animal i (Borchers and Efford 2008). The sums are over all individuals ever seen. Session-specific abundances are inferred by distributing N or D over sessions according to the entry probabilities b . Alternatively (HTbysession = TRUE) the H-T estimate may be based on the number detected in each session and the corresponding session-specific estimates of p or a .

2. Any of the recruitment parameters in Table 2 or Table 4 may be computed from any other model of the same class (non-spatial or spatial)¹².

Both goals are served by the `derived` method for `openCR` objects. Among other outputs, this generates a summary table with point estimates of all relevant parameters. We demonstrate this with a new dipper model, fitted using conditional likelihood:

```
dipperCL <- openCR.fit(dipperCH, type = 'JSSA1CL',
  model = list(lambda~t, phi~t))
# only these parameters are in the model and estimated directly,
names(predict(dipperCL))
```

```
## [1] "p"      "phi"     "lambda"
# but we can derive b, f, gamma and N, as well as the super-population N
d <- derived(dipperCL)
print(d, digits = 3, legend = TRUE)
```

```
## Total number observed 294
## Parameters in model p, phi, lambda
## Superpopulation size 310.6
## Session-specific counts and estimates:
##
## session t n R m r z time p phi lambda b f gamma kappa N
## 1 1 22 22 0 13 0 0 0.902 0.626 2.792 0.0785 2.166 NA NA 24.4
## 2 2 60 60 11 25 2 1 0.902 0.454 1.265 0.1701 0.811 0.224 2.23 68.1
## 3 3 78 78 26 36 1 2 0.902 0.478 1.026 0.1778 0.548 0.359 2.36 86.2
## 4 4 80 80 35 48 2 3 0.902 0.624 1.104 0.1519 0.480 0.466 2.05 88.4
## 5 5 88 88 47 51 3 4 0.902 0.608 1.103 0.1365 0.495 0.566 1.86 97.6
## 6 6 98 98 52 52 2 5 0.902 0.583 0.958 0.1554 0.375 0.551 2.09 107.6
## 7 7 93 93 54 0 0 6 0.902 NA NA 0.1298 NA 0.609 1.77 103.1
##
## Field Definition
## -----
## session primary session
## t primary session
## n number observed
## R number released
## m number already marked
## r number recaptured in later session
## z number known alive but not caught
## time accumulated time since start
## p detection probability per secondary session
## phi apparent survival per unit time
## lambda population growth rate per unit time
## b entry probabilities
## f per capita recruitment per unit time
## gamma seniority (cf reverse-time phi)
## kappa recruitment parameter of Link and Barker (2005)
```

¹²However, the effect of a constraint (e.g., parameter constant over sessions) will vary depending on the parameter to which it is applied.

```
## N      population size
```

The `print` method for objects from `derived` provides some control over formatting, as shown. Use the `Dscale` argument to change area units (spatial models only).

`derived` does not yet provide delta-method SE or confidence intervals for derived parameters. A reliable workaround for abundance parameters (N, D)¹³ is to (i) infer the point estimates with `derived`¹⁴, (ii) assemble a start vector on the link scale(s) for an equivalent full-likelihood `openCR.fit` model that includes the derived abundances, and (iii) run `openCR.fit` with `method = "none"` to compute the hessian at the MLE, and hence the full variance-covariance matrix.

10 Simulating open-population data

The `seccr` functions `sim.popn` and `sim.caphist` provide the means to generate spatial open-population data with known survival probability, population trend λ and detection parameters. Open population data are generated by setting `nsessions > 1` in `sim.popn` and specifying a value for λ . Turnover settings are controlled by components of the ‘details’ argument of `sim.popn`. The `seccr` help page `?turnover` should be consulted. `sim.caphist` should be called with `renumber = FALSE` (otherwise individual capture histories cannot be matched across primary sessions).

Use the `openCR` function `sim.nonspatial` to generate non-spatial open-population data. `openCR` also provides these functions to streamline simulation and speed it up by using multiple cores –

Function	Purpose
<code>runsim.nonspatial</code>	Generate data with <code>sim.nonspatial</code> and fit models using <code>openCR.fit</code>
<code>runsim.spatial</code>	Generate data with <code>sim.popn</code> and <code>sim.caphist</code> , and fit models using <code>openCR.fit</code>
<code>sumsims</code>	Summarise list output from <code>runsim.nonspatial</code> or <code>runsim.spatial</code>

`runsim.nonspatial` and `runsim.spatial` are essentially wrappers; the user must provide appropriate argument values for each of the nested functions. See `openCR-simulations.pdf` for example code.

11 Troubleshooting

11.1 Nonidentifiability

It is common for some session-specific parameters of open capture–recapture models to be nonidentifiable, either for structural reasons or because the particular dataset is uninformative (e.g., Gimenez et al. 2004).

The main diagnostic is the rank of the Hessian matrix. If the rank is less than the number of parameters then the model is not fully identifiable and the estimates of some parameters will be confounded or unreliable. Matrix rank is determined numerically by counting non-zero eigenvalues. Computed eigenvalues of non-identifiable parameters may appear as small positive numbers, so it is necessary to apply an arbitrary numerical threshold.

Exactly which parameter estimates are unreliable can usually be discerned from computed variances (SE and confidence intervals). Data cloning (Lele et al. 2010) is also helpful; function `cloned.fit` implements the method for nonspatial models.

¹³This may sometimes be feasible for derived recruitment parameters, but given the doubts introduced by differing constraints it is better just to refit the model.

¹⁴These are also the MLE when `distribution = "poisson"` (e.g., Schofield and Barker 2016).

Session-specific turnover parameters may become nonidentifiable if home ranges are allowed to move freely between primary sessions (`movementmodel = 'uncorrelated'`). Intuitively, this is because radical changes in individual detection probability (due to proximity to detectors) cannot be separated from mortality and recruitment.

11.2 Failure of numerical maximization

Bad estimates (zero, very large, close to starting values or zero variance) may merely indicate a problem with the maximization algorithm rather than nonidentifiability.

11.2.1 Starting values

Numerical maximization of the likelihood requires appropriate starting values for the parameters. If starting values are poor then initial evaluations of the likelihood may return an infinite value, or otherwise provide inadequate direction for the numerical algorithm.

`openCR.fit` provides a mechanism for recycling earlier estimates as starting values: simply provide the name of a previously fitted model as the `start` argument. Parameters shared between the models will be set to the old estimates, while unmatched parameters will be set to defaults. A list of two previous models may be provided; values from the first take precedence.

11.2.2 Boundary estimates

Variance estimation based on the Hessian matrix fails if the estimate lies on a boundary of the parameter space. Computed SE are then extreme, and confidence limits are implausible. This commonly happens when apparent survival (ϕ) approaches 1.0. Boundary estimates are more benign than other reasons for failure (the estimates themselves may be reliable). Alternative methods for variance estimation in this case have not been implemented.

Using the “sin” link for parameters bounded by 0 and 1 (the probability parameters p and ϕ) can be helpful.

11.2.3 Alternative algorithms

The default method for maximizing the likelihood function is Newton-Raphson as implemented in the R function `nlm`. This relies on numerical gradient estimates, which can cause trouble. Avoid gradient estimation entirely by using the somewhat slower ‘Nelder-Mead’ method of function `optim` e.g.,

```
fitnr <- openCR.fit(ovenCH, type = 'JSSA1CL', model = list(phi ~ t, lambda~t))
fitnm <- openCR.fit(ovenCH, type = 'JSSA1CL', model = list(phi ~ t, lambda~t),
                   method = "Nelder-Mead", details = list(control = list(maxit = 5000)))
```

The default maximum number of likelihood evaluations for the Nelder-Mead algorithm (500) is often too small and results in a “probable maximization error” warning. Here we increase it to 2000 by setting the `details` argument “control” that is passed to `optim`.

Somewhat alarmingly, the NM algorithm settles on a lower log likelihood and different estimates:

```
AIC(fitnm,fitnr)
```

```
##                model npar rank logLik  AIC AICc  dAIC  AICwt
## fitnr p~1 phi~t lambda~t    9    9 -656.7 1331 1334 0.000 0.9898
## fitnm p~1 phi~t lambda~t    9    9 -661.3 1341 1344 9.154 0.0102
```

We can fix that by feeding Nelder-Mead the starting values from another model:

```
fitnm <- openCR.fit(ovenCH, type = 'JSSAICL', model = list(phi ~ t, lambda~t),
  method = "Nelder-Mead", details = list(control = list(maxit = 2000)),
  start = fitnr)
AIC(fitnm,fitnr)
```

```
##                model npar rank logLik  AIC AICc dAIC AICwt
## fitnm p~1 phi~t lambda~t    9   9 -656.7 1331 1334    0  0.5
## fitnr p~1 phi~t lambda~t    9   9 -656.7 1331 1334    0  0.5
```

In the longer term, better maximizers are needed.

11.3 Speed

Spatial models are slow to fit. Consider these options

- Use no more mask points than necessary. Typically about 1000 will do (may not apply for kernel movement models).
- Data with many occasions (secondary sessions) should be collapsed.
- If you are interested in several models, fit them all at once with `par.openCR.fit` and `ncores > 1`. This can save a lot of time.
- Don't assume that increasing the number of cores will necessarily speed up computations; additional cores impose overheads in cluster setup and communication.
- Use the conditional likelihood (CL) models: estimates of phi and lambda are often all you need, and `derive` can give estimates of abundance (superN, N, superD, and D) from CL models, as well as alternative measures of recruitment.
- Avoid individual covariates with many levels. This applies especially to continuous individual covariates: normally these should be discretized (coded as a few ordered categories, but *not* converted to factor).
- First fit the most simple model and add complexity, using a simpler related model for 'start'.
- 'secr' data with detector type 'multi' fit much faster than 'proximity' data; use this option if it makes sense (and even maybe when it doesn't).
- For problems with many parameters, 'cyclic fixing' may be useful (Schwarz and Arnason 1996; Pledger et al. 2003).

`openCR` \geq 1.2 is generally faster than `openCR` 1.1 because it automatically uses multiple threads to run some calculations in parallel. The speed difference can be impressive - around 4x. Multithreading uses `RcppParallel`. A couple of tuning parameters are available. The number of threads is set with `ncores` argument in `openCR.fit`. By default, `openCR` uses one less than the number of (virtual) cores available (i.e. 7 on a quad-core desktop with hyperthreading). The 'grain size' (see `RcppParallel` may be varied with `details$grain`, but seems to have little effect.

12 Extras

12.1 Sampling variance warning

Full models (not CL or Pradel) include superpopulation size N as a variable. The default in `openCR` for both non-spatial and spatial models is to treat N as a Poisson variable, from which it follows that the number of individuals detected at least once (n) is also Poisson. This is also the default in `secr`. However, estimates from POPAN models in MARK treat N as fixed and n as binomial. The assumption of fixed N leads to narrower confidence intervals and estimates of detection and turnover parameters that differ slightly from conditional likelihood models (see e.g. Schofield and Barker 2016). To obtain JSSA estimates from `openCR` that match those from MARK it is necessary to set `distribution = "binomial"`.

12.2 Example datasets

Several examples of analyses with **openCR** are given in the associated vignette `openCR-examples.pdf`. These use data already formatted as **seccr** capthist objects in R; the objects are provided in one or other package. All are available immediately **openCR** is loaded with `library`. Each has its own help page.

Table 6. Data objects in **openCR**. ‘RD’ indicates robust design with multiple secondary sessions. See `openCR-examples.pdf` for references.

Data object	Spatial	RD	Species etc.	Source
microtusCH etc.	No	Yes*	Meadow vole <i>Microtus pennsylvanicus</i> USA	Nichols et al. (1984), Williams et al. (2002)
FebpossumCH	No	Yes	Brushtail possum <i>Trichosurus vulpecula</i> New Zealand	M. Efford unpubl.
dipperCH	No	No	European dipper <i>Cinclus cinclus</i> France	Lebreton et al. (1992), MARK
gonodontisCH	No	No	Moth <i>Gonodontis bidentata</i> England	Bishop et al. (1978), Crosbie (1979)
fieldvoleCH	Yes	Yes	Field vole <i>Microtus agrestis</i> Norway	Ergon and Lambin (2013)

Table 7. Multi-session data objects in **seccr**.

Data object	Spatial	RD	Species etc.	Source
OVpossumCH	Yes	Yes	Brushtail possum <i>Trichosurus vulpecula</i> New Zealand	M. Efford unpubl.
ovenCHp	Yes	Yes	Ovenbird <i>Seiurus aurocapilla</i> USA	D. Dawson and M. Efford unpubl.

12.3 Testing assumptions

This is generally an undeveloped field for spatially explicit capture–recapture models. Demonstrating that assumptions were not satisfied may also be of no consequence: we would usually ignore such a finding if the estimator is reasonably robust.

For Cormack–Jolly–Seber (nonspatial) models there is an established suite of tests following Burnham et al. (1987). The tests have been implemented in the U-CARE software of Choquet et al. (2009), recently translated into R by Gimenez et al (2018) as package **R2ucare**. Program RELEASE (Burnham et al. 1987) also implements the core CJS tests and is available through MARK.

The **openCR** function `ucare.cjs` is a wrapper for relevant functions in **R2ucare**, which should be installed. We briefly demonstrate it here for the dipper data of Marzolin (1988).

```
if (requireNamespace("R2ucare"))
  ucare.cjs(dipperCH, verbose = FALSE, by = 'sex')
```

```
## Loading required namespace: R2ucare
## $Male
## $Male$components
##      stat df p_val sign_test
## test3sr 6.778 5 0.238 -1.530
## test3sm 0.000 2 1.000      NA
```

```

## test2ct 4.284 2 0.117 -1.035
## test2cl 0.000 0 1.000 NA
##
## $Male$overall_CJS
##               chi2 degree_of_freedom p_value
## Gof test for CJS model: 11.06          9 0.271
##
##
## $Female
## $Female$components
##      stat df p_val sign_test
## test3sr 4.985 5 0.418 1.428
## test3sm 2.041 3 0.564 NA
## test2ct 3.250 4 0.517 -0.902
## test2cl 0.000 0 1.000 NA
##
## $Female$overall_CJS
##               chi2 degree_of_freedom p_value
## Gof test for CJS model: 10.28        12 0.592

```

This invocation of `ucare.cjs` calls the **R2ucare** functions `test3sr`, `test3sm`, `test2ct`, `test2cl` and `overall_CJS` for each sex and provides a condensed report. For interpretation see the original papers, the **R2ucare** vignette, and Chapter 5 of the MARK book (Cooch and White 2017). Lebreton et al. (1992: 86) indicate only Test 3SR is meaningful for these data (see also `openCR-examples.pdf`).

12.4 Limitations of openCR

`openCR` does not do

1. Continuous random effects (consider finite mixtures as an alternative)
2. Parameter counting to adjust AIC
3. Overdispersion adjustment (chat, QAIC) or goodness-of-fit tests, except for `ucare.cjs` (above).
4. MCMC
5. Bootstrap confidence intervals
6. Temporary emigration parameterizations of non-spatial robust-design models
7. Age-specific survival curves (Weibull etc.)
8. Mark-resight
9. SE for derived parameters and estimates with mlogit link (to be fixed)

Parameter counting and overdispersion adjustment are probably the most critical omissions. See Cooch and White (2017) for detailed coverage in the context of MARK.

12.5 Differences from secr

Defaults for some arguments differ between `openCR.fit` and `secr.fit`. For `openCR.fit` –

1. `trace = FALSE`
2. By default the reported log likelihood and AIC do not include the multinomial constant (`details$multinom = FALSE`)
3. The default criterion for AIC.openCR is ‘AIC’, not ‘AICc’ as in `secr`.

`distribution` has been elevated to a full argument rather than merely a component of `details`. This argument describes the distribution of the number of individuals detected (default distribution = “poisson”) (see here).

When `details$LLonly = TRUE`, `openCR.fit` returns a vector with the log likelihood in position 1, followed by the named starting values of the coefficients (beta parameters) (`secr.fit` returns only the log likelihood).

In `secr` the argument `CL` is used in `secr.fit` to switch between full- and conditional-likelihood models. In `openCR` conditional-likelihood models are given a separate `type` with the suffix `CL`.

The predictor ‘t’ is used in `secr` models to indicate a factor with one level for each *secondary* session. In `openCR` it is a synonym for ‘session’, i.e. a factor with one level for each *primary* session. This is consistent with the use of ‘t’ in Lebreton et al. (1992) and makes for more compact model specification. In the unlikely event that you want to code a model with one level for each secondary session in `openCR`, use the ‘timecov’ argument.

Parts of `openCR` are coded in C++, via the R package `Rcpp`, whereas `secr` uses C. The `Rcpp` interface requires less copying of data, and enables the use of multiple threads via `RcppParallel`. `openCR` also duplicates some C++ functions in native R code, which is useful for debugging. Select the R version by setting `details = list(R = TRUE)` in `openCR.fit`. This currently works for most models except those with detector type ‘multi’.

These features of `secr` are not available in `openCR`

1. Hybrid mixture models (`hcov` in `secr`)
2. Groups (use `CL` and individual covariates, or see `marked`)
3. Regression splines from `mgcv`
4. Model averaging
5. Density surfaces and other spatial density models
6. Post-hoc probability density of activity centres (`fxi` in `secr`)
7. Non-point detectors (polygon, polygonX etc. in `secr`)
8. ‘collate’ function (`make.table` may do the job)
9. Variable effort for *nonspatial* models (cf Efford, Borchers and Mowat 2013) (The ‘usage’ attribute of traps objects is applied in spatial `openCR` models).
10. Negative binomial counts (`binomN<0`)

12.6 Relationship to other software

The non-spatial capability of `openCR` largely duplicates MARK and RMark. Several of the nonspatial model types have exact matches in MARK (Table 8).

Table 8. Relationship of non-spatial `openCR` models to MARK model types

<code>openCR</code> type	MARK model	Reference
CJS	CJS	Seber (1982)
JSSAb	POPAN	Schwarz and Arnason (1996)
JSSAfCL	LinkBarker	Link and Barker (2005)
Pradel	Pradlambda	Pradel (1996)
Pradelg	Pradsen	Pradel (1996)

The R package `marked` (Laake, Johnson and Conn 2013) also overlaps substantially with the non-spatial features of `openCR`. Its interface echoes `RMark` just as `openCR` echoes `secr`. `marked` has some fancy features for individual covariates and random effects, and promises fast processing of large datasets. `marked` 1.1.13 includes full-likelihood JSSA (POPAN) models parameterized in terms of entry probabilities (type JSSAb)¹⁵, but not the other JSSA options in Table 2.

¹⁵dipper example in `openCR-examples.pdf`.

13 References

- Borchers, D. L. and Efford, M. G. (2008) Spatially explicit maximum likelihood methods for capture–recapture studies. *Biometrics* **64**, 377–385.
- Burnham, K. P., Anderson, D. R., White, G. C., Brownie, C. and Pollock, K. H. (1987) *Design and Analysis Methods for Fish Survival Experiments Based on Release-Recapture*. American Fisheries Society Monograph 5. Bethesda, Maryland, USA.
- Chandler, R. B. and Clark, J. D. (2014) Spatially explicit integrated population models. *Methods in Ecology and Evolution* **5**, 1351–1360.
- Choquet, R., Lebreton, J.-D., Gimenez, O., Reboulet, A.-M. and Pradel, R. (2009) U-CARE: Utilities for performing goodness of fit tests and manipulating CApture-REcapture data. *Ecography* **32**, 1071–1074.
- Nathan, R., Klein, E., Robledo-Arnuncio, J. J. and Revilla, E. (2012) Dispersal kernels: a review. In: J. Clobert et al. *Dispersal Ecology and Evolution*. Oxford University Press. Pp 187–210.
- Cooch, E. and White, G. (eds) (2017) *Program MARK: A Gentle Introduction*. 17th edition. Available online at <http://www.phidot.org/software/mark/docs/book/>.
- Crosbie, S. F. and Manly, B. F. J. (1985) Parsimonious modelling of capture–mark–recapture studies. *Biometrics* **41**, 385–398.
- Efford, M. G. (2004) Density estimation in live-trapping studies. *Oikos* **106**, 598–610.
- Efford, M. G. (2019). secr: Spatially explicit capture–recapture models. R package version 3.2.1. <https://CRAN.R-project.org/package=secr>
- Efford, M. G., Borchers D. L. and Byrom, A. E. (2009) Density estimation by spatially explicit capture–recapture: likelihood-based methods. In: D. L. Thomson, E. G. Cooch, M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer. Pp 255–269.
- Efford, M. G., Borchers D. L. and Mowat, G. (2013) Varying effort in capture–recapture studies. *Methods in Ecology and Evolution* **4**, 629–636.
- Efford, M. G. and Fewster, R. M. (2013) Estimating population size by spatially explicit capture–recapture. *Oikos* **122**, 918–928.
- Efford, M. G. and Schofield, M. R. (2019). A spatial open-population capture—recapture model. *Biometrics* (In revision).
- 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.
- Fletcher, D. J. (2012) Estimating overdispersion when fitting a generalized linear model to sparse data. *Biometrika* **99**, 230–237.
- Gardner, B. J., Reppucci, J., Lucherini, M. and Royle, J. A. (2010) Spatially-explicit inference for open populations: estimating demographic parameters from camera-trap studies. *Ecology* **91**, 3376–3383.
- Gimenez, O., Viallefont, A., Catchpole, E. A., Choquet, R. and Morgan, B. J. T. (2004) Methods for investigating parameter redundancy. *Animal Biodiversity and Conservation* **27**, 561–572.
- Gimenez, O., Lebreton, J.-D., Choquet, R. and Pradel, R. (2017) R2ucare: Goodness-of-Fit Tests for Capture-Recapture Models. R package version 1.0.0. <https://CRAN.R-project.org/package=R2ucare>
- Gimenez, O., Lebreton, J.-D., Choquet, R. and Pradel, R. (2018) R2ucare: An R package to perform goodness-of-fit tests for capture–recapture models. *Methods in Ecology and Evolution* **9**, 1749–1754.
- Glennie, R., Borchers, D. L., Murchie, M. Harmsen, B. J. and Foster, R. J. (2019) Open population maximum likelihood spatial capture–recapture. *Biometrics* In press.

- Hines, J. E., Kendall, W. L. and Nichols, J. D. (2003) On the use of the robust design with transient capture–recapture models. *The Auk* **120**, 1151–1158.
- Huggins, R. M. (1989) On the statistical analysis of capture experiments. *Biometrika* **76**, 133–140.
- 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.
- Laake, J. and Rexstad E. (2014) Appendix C. RMark - an alternative approach to building linear models in MARK. In: Cooch, E. and White, G. (eds) *Program MARK: A Gentle Introduction*. 13th edition. <http://www.phidot.org/software/mark/docs/book/>.
- 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.
- Lele, S.R., Nadeem, K. and Schmuland, B. (2010) Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association* **105**, 1617–1625.
- 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 cotes de Lorraine. L'Oiseau et la Revue Francaise d'Ornithologie 58:277-286.
- Nathan , R., Klein, E., Robledo-Arnuncio, J. J. and Revilla, E. (2012) Dispersal kernels: a review. In: J Clobert et al. *Dispersal Ecology and Evolution*. Oxford University Press. Pp 187–210.
- 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.
- Nichols, J. D. (2016) And the first one now will later be last: time-reversal in Cormack–Jolly–Seber models. *Statistical Science* **31**, 175–190.
- Pledger, S., Pollock, K. H. and Norris, J. L. (2003) Open capture–recapture models with heterogeneity: I. Cormack–Jolly–Seber model. *Biometrics* **59**, 786–794.
- 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. (1982) A capture–recapture design robust to unequal probability of capture. *Journal of Wildlife Management* **46**, 752–757.
- Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* **52**, 703–709.
- Pradel, R., Hines, J. E., Lebreton, J.-D. and Nichols, J. D. (1997) Capture–recapture survival models taking account of transients. *Biometrics* **53**, 60–72.
- Royle, J. A., Chandler, R. B., Sollmann, R. and Gardner, B. (2014) *Spatial capture–recapture*. Academic Press.
- Schwarz, C. J. (2001) The Jolly-Seber model: more than just abundance. *Journal of Agricultural, Biological, and Environmental Statistics* **6**, 195–205.
- 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.
- White, G. C. and Burnham, K. P. (1999) Program MARK: survival estimation from populations of marked animals. *Bird Study* **46**, Supplement S120–S139.
- Whittington, J. and Sawaya, M. A. (2015) A comparison of grizzly bear demographic parameters estimated from non-spatial and spatial open population capture–recapture models. *PLoS ONE* **10**, e0134446. <https://doi.org/10.1371/journal.pone.0134446>
- Williams, B. K., Nichols, J. D. and Conroy, M. J. (2002) *Analysis and management of animal populations*. Academic Press, San Diego.