oSCR : : CHEAT SHEET

The oSCR package, pronounced "Oscar", provides a set of functions for working with Spatial Capture Recapture (SCR) models.

Getting the package

Package hosted on GitHub

```
library (devtools)
install_github("jaroyle/oSCR")
library(oSCR)
```

Workflow

- Every model you run on oSCR has the following 4 basic steps.
- Modeled after unmarked workflow

1. Format the sampling data

One file for each one:

- Spatial encounter histories
- Detector information

2. Define and format the State Space

- Size and resolution of the *state space*
- Spatial covariates for density

3. Analyze the data – model fitting

- Likelihood based: use AIC to do model selection
- No need to use other packages, oSCR has helper functions to do the model selection.

4. Post processing model output for inference:

- This means that now that you have your parameters all you have to do is interpret your results!

Modelling framework

A. Single-session models

- Repeated sample occasions on a single population of individuals using a single array of traps.

B. Multi-session models

- Data grouped in strata or groups which are independent in space or time.

C. Explicit sex-structured models

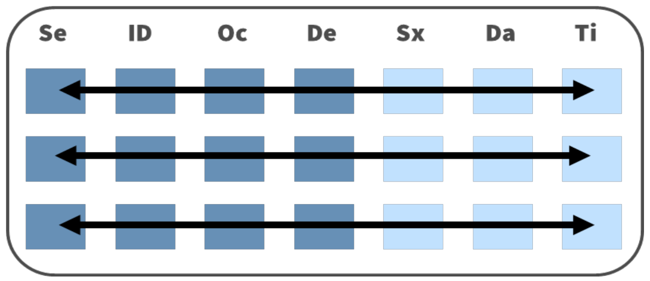
D. Multi-session sex-structured models

1. Format sampling data

Before starting to use oSCR you need to format the datafiles in a scrFrame which consists of two basic spreadsheets: **edf** and **tdf**.

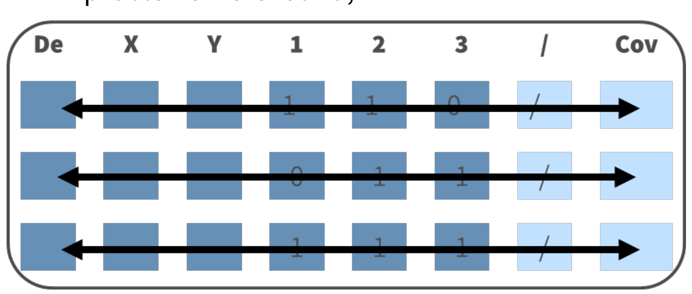
1.1 edf : encounter data file.

- Single **data frame**.
- Each row has individual detection events.
- Dark blue = required; light blue = optional.
- Columns contain capture information:
  - Session (Se)
  - Individual ID (ID)
  - Occasion (Oc)
  - Detector* (De)
  - Sex (Sx)
  - Date (Da)
  - Time (Ti)



1.2 tdf : trap deployment data file.

- A **list** with information for each session (tdf1, tdf2, …).
- Each row is a trap.
- Columns contain trap information
  - Detector* (De)
  - X (required, UTM)
  - Y (required, UTM)
  - Binary trap operation data for malfunctions, rotations (required if problems were found; 1, 2, 3, … n)
  - Separator (e.g., /)
  - Trap level covariates (different column per covariate)



*Notice that both edf and tdf have the same **Detector (De)** column that **MUST** match (same name, class, relational database).

1.3 data2oscr() : is a function that links **edf** and **tdf** files via the detector* names. Creates **scrFrame**.

```
data <- data2oscr(
edf,            # encounter data file
tdf,            # list containing trap deployment file
sess.col*,      # session col number or name in edf
id.col*,        # individual ID col # or name in edf
occ.col,        # occasion col number or name in edf
trap.col*,      # detector col number or name in edf
sex.col*,       # sex col number or name in edf
sex.nacode,     # character for unknown sex in edf
K,              # number of occasions
ntraps,         # number of traps
trapcov.names,  # vector of un-numbered cov names
tdf.sep)        # separator (e.g., "/")
```

```
* which(colnames(edf) %in% "name of column in edf")
```

1.4 Summary functions for scrFrame :

- scrFrame contains information from the **edf** and **tdf** via detector names.

```
sf<-data$scrFrame
```

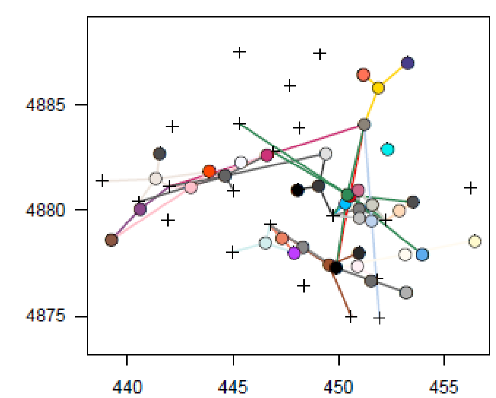
sf$caphist	Array of individual-by-trap-by-occasion (n x J x K). Binary or counts.
sf$traps	Data frame containing at least trap ID and coordinates of traps. Best with UTM.
sf$indcovs	Sex data (0 female, 1 male) or any bivariate covariate. NAs allowed.
sf$trapCovs	List of session specific trap covariates. Row per trap, and column per covariate.
sf$sigCovs	A data frame of covariates that affect space use (sigma, σ).
sf$trapOperation	A list of session specific information on trap operational data.
sf$occasions	A vector of number of occasions per session .
sf$mmdm	Mean maximum distance moved pooled across sessions. ½ mmdm ~ σ
sf$mdm	Maximum distance moved pooled across sessions.
$telemetry	Telemetry object for fitting resource selection models.

1.5 Summary of scrFrame

```
sf
                   S1
n individuals      47
n traps            38
n occasions         8

                   S1
avg caps         3.21
avg spatial caps 2.02
mmdm             4.65
```

1.6 Spatial captures per session

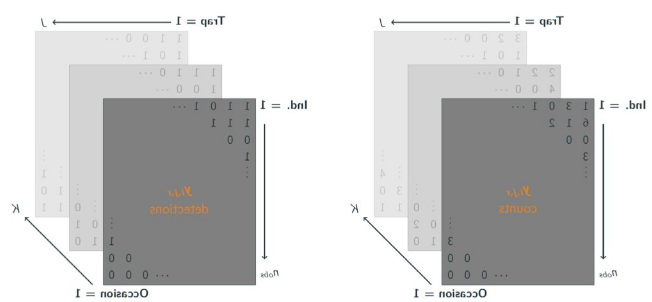
```
plot(sf) #y and x are UTM
```



Package created by Chris Sutherland • csutherland@umass.edu • Learn more at https://sites.google.com/site/spatialcapturerecapture/oscr-package • oSCR version 0.42.0 •

oSCR : : CHEAT SHEET

1.4.1 Navigating the scrFrame



Capture history

- Session 1, all individuals, all traps, occasion 3
  `sf$caphist[[1]][ , ,3]`
- Session 1, individual 4, all traps, all occasions
  `sf$caphist[[1]][4, , ]`

Traps

- Session 1 trap coordinates
  `sf$traps[[1]]`

Trap covariates

- Trap covariate df session 1 occasion 4
  `sf$trapCovs[[1]][[4]]`

Trap operation

- Session 1 trap trap operation matrix
  `sf$trapOperation [[1]]`

Covariates that affect sigma (σ)

- These covariates are NOT session specific. This is a sessions=rows dataframe
  `sf$ sigCovs[[1]]`

Vectors and single numbers

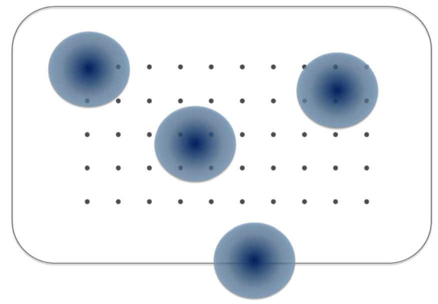
```
sf$ occasions
sf$mmdm
sf$mdm
```

Datasets available

> data(package = “oSCR”)
> data(ocelot)
> data(“beardata”)
> data(“nybears”)
> data(“peromyscus”)
> data(“mink”)

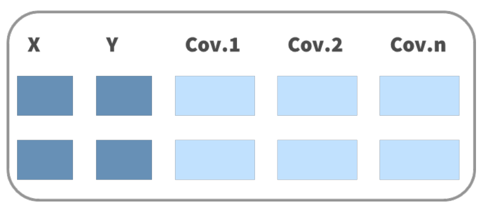
2. Create the State Space

The **State Space (S)** is the core element of SCR models. It defines where individuals can live and should represent activity centers of all sampled individuals.



ssDF: the State Space Data Frame

- List with spatially explicit information from each session.
- At least include the coordinates (X, Y) of the discrete state space (UTM).
- Can include spatial covariates for a continuous state space to study variation in Density.
- Non habitat can be removed by removing unwanted coordinates (e.g., parking lot).



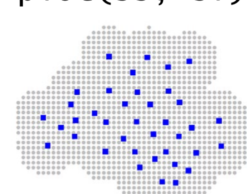
2.1. make.ssDF():

- Remember that ½ mmdm ~ σ
- Extracts covariates and removes non habitat

```
ss <- make.ssDF(scrFrame,
buffer, #~3 to 4σ around traps
res) # ≤ σ̂
```

2.2. Plot the state space

- Plot state space `plot(ss)`
- Plot state space & traps `plot(ss, sf)`

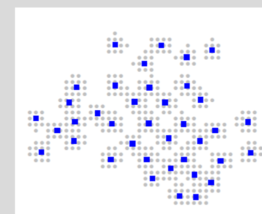


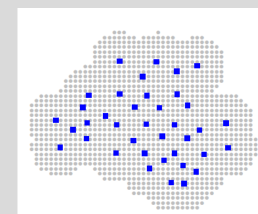
Vary the buffer and/or resolution

Varying buffer, fixed resolution

```
make.ssDF(sf,
buffer = 1,
res = 0.5)
```

```
make.ssDF(sf,
buffer = 3,
res = 0.5)
```

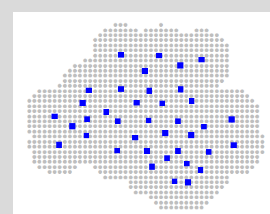


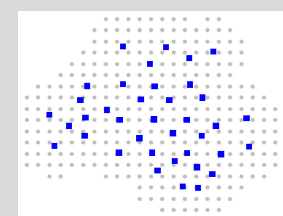


Fixed buffer, varying resolution

```
make.ssDF(sf,
buffer = 3,
res = 0.1)
```

```
make.ssDF(sf,
buffer = 3,
res = 0.5)
```





3. Fit the model

3.1. Single-session model: Fit the model with oSCR.fit():

```
sf <- data$scrFrame
mod <- oSCR.fit(model,
                scrFrame, #sf
                ssDF, …)
```

- See pg. 3 for null model and multi-session models.

model is a list with 3 basic formulations:
`list(D ~ 1, p0 ~ 1, sig ~ 1)`

Variation in…	
D	pixel density
p0	baseline encounter prob/rate
sig	sigma (σ)

3.2. Backtransform to the real scale

```
get.real(model,
         newdata,
         d.factor,
         type)
```

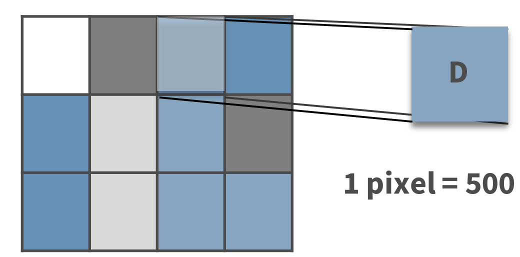
model	fitted model
newdata	Optional new data object for predictions
d.factor	optional scale the estimates to a different resolution
type	density (“dens”), detection probability (“det”), sigma(“sig”)

“dens”
Sex-specific estimates of density, and the density estimates are **per pixel.**

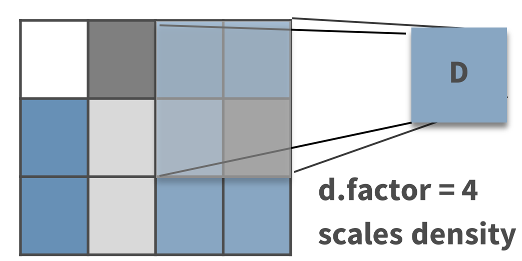
“det”
Estimate of detection at distance from activity center = 0.

“sig”
Estimates of the spatial scale of detection.

d.factor



1 pixel = 500 m x 500 m



d.factor = 4
scales density to 1 km²
When using the 500 m resolution

oSCR : : CHEAT SHEET

Page 3 describes the specific functions and workflow for the null model and multi-session model in the oSCR package.

Model specifics

Null model (SCR_0)

- The null model assumes homogeneous density which means all pixels have the same expected density.
- For additional arguments see `?oSCR.fit()`

```
mod1 <- oSCR.fit(list(D ~ 1,
p0 ~ 1, sig ~ 1),
scrFrame, #scrFrame object
ssDF, #ssDF object
… ) #other arguments
mod1 #summary
```

If you included sex as a covariate in the scrFrame:

- Sex ratio psi () will be included in the summary
- Can compare AIC with and without sex effects

Multi-session model

Are your data organized in multi-sessions and you want to analyze all of them jointly?

Spatial sessions: different study areas (e.g., parks, trapping grids)

Temporal sessions: same areas different times (e.g. seasons, years)

Session specific **population size** N_g (g=group/session)

- Test for differences among sessions using AIC.
- Can share parameters among sessions or not.

- The **multi-session** model follows similar steps as the single session model.
- The **edf** files from multiple sessions may be merged into one data frame prior to data2oscr

  edf <- rbind(edf1, edf2, …)
- The **tdf** files must be separate files for each session.

1. data2oscr for multi-session scrFrame

```
data <- data2oscr(
 edf,   # include session column
 list(tdf1, tdf2, …), # tdf files
 sess.col*,  # session col in edf
 id.col*,  # individual ID col in edf
 occ.col,  # occasion col in edf
 trap.col*,  # detector col in edf
 sex.col*,  # sex col in edf
 sex.nacode,  # unknown sex in edf
 K,  # vector with occasions per session
 ntraps)  # vector with traps per session

sf <- data$sf
sf # summary info per session (S1, S2..)
```

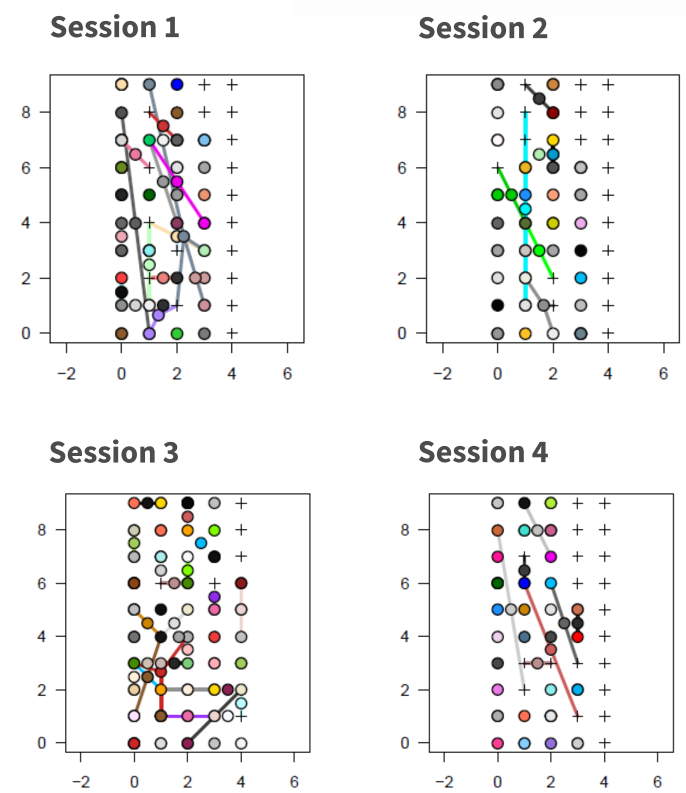
1.2. Summary of multi-session scrFrame

```
              S1  S2   S3  S4
n individuals 77  60  108  54
n traps       50  50   50  50
n occasions    7   5    6   4

                    S1   S2   S3   S4
avg caps          1.91 1.47 1.71 1.37
avg spatial caps  1.30 1.15 1.27 1.13
mmdm              2.57 2.32 1.76 2.84

Pooled MMDM:  2.21
```

1.3. Plot spatial captures in a multi-session scrFrame

- Use plot(sf) to plot a spatial capture per session

```
par(mfrow=c(1,n)) # n = sessions
plot(sf) # plot all sessions
```



2. Make the State Space Data Frame of a multi-session scrFrame

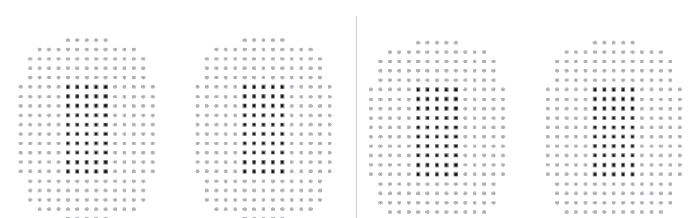
```
ss <- make.ssDF(
scrFrame, # multi-session
buffer, #buffer width
res) #state space resolution
```

- You can vary the buffer and resolution as in the single-session model.

```
?make.ssDF() # Look at the help file for other arguments
```

- Visualize the state space

```
par(mfrow=c(1,n)) # n = sessions
plot.ssDF(ss, # state space
          sf) # traps
```



3. Model fitting

- Specify models that consider or not variation among sessions.
  - fixed vs. session specific **D**
  - fixed vs. session specific **p0**
  - fixed vs. session specific **space use (σ)**

Model	Algebra	In `oSCR.fit`
Density	$log(D(s_i)) = \beta_0$	`D ~ 1`
Density	$log(D(s_i)) = \beta_0 + \beta_{1(g)} Session_g$	`D ~ session`
Detection	$logit(p_0) = \alpha_0$	`p0 ~ 1`
Detection	$logit(p_0) = \alpha_0 + \alpha_{1(g)} Session_g$	`p0 ~ session`
Space use	$log(\sigma) = \gamma_0$	`sig ~ 1`
Space use	$log(\sigma) = \gamma_0 + \gamma_{1(g)} Session_g$	`sig ~ session`

- Include all models into a list using fitList.oSCR():

```
f1 <- fitList.oSCR(
mods, # list of fitted models
rename) # if TRUE models are renamed with sensible names
```

- Compare multiple models

```
ms <- modSel.oSCR(f1)
```

- Generate an AIC table to compare multiple models

```
ms$aic
```

- Generate a coefficient table

```
ms$coef.tab
```

- Generate a model averaged coefficients

```
ma <- ma.coef(ms) # include a modSel.oSCR object
```

3.1. Back transform to the real scale

```
top.model <- m3

pred.df <- data.frame(session = factor (c(1, 2, 3, 4, … )))

pred.det <- get.real(
model = top.model, type = “det”,
newdata = pred.df)
```