Ecography

ECOG-04551
Sutherland, C., Royle, J. A. and Linden, D. W. 2019. oSCR: a spatial capture–recapture R package for inference about spatial ecological processes. – Ecography doi: 10.1111/ecog.04551

Appendix for the R package oSCR

The RBS worked example

The script below implements exactly the redback salamnder (*P. cinereus*) example analysis described in the manuscript.

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

```
library(oSCR)
data(rbs_ecography)
ls()
[1] "rbs.edf"  "rbs.tdf1" "rbs.tdf2" "rbs.tdf3" "rbs.tdf4"
```

The data consists of effort data contained in the session-specific trap deployment files (TDF) (session = cover object array). Note that, trap characteristics (name and coordinates) and the four covariates, are separated by a column of '/"s:

```
str(rbs.tdf4)
'data.frame':   50 obs. of  16 variables:
 $ board  : Factor w/ 50 levels "A1","A10","A2",..: 1 3 4 5 6 7 8 9 10 2 ...
 $ X      : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Y      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sep    : Factor w/ 1 level "/": 1 1 1 1 1 1 1 1 1 1 ...
 $ day.1  : int  256 256 256 256 256 256 256 256 256 256 ...
 $ day.2  : int  284 284 284 284 284 284 284 284 284 284 ...
 $ day.3  : int  304 304 304 304 304 304 304 304 304 304 ...
 $ day.4  : int  311 311 311 311 311 311 311 311 311 311 ...
 $ jday.1 : num  1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 ...
 $ jday.2 : int  4 4 4 4 4 4 4 4 4 4 ...
 $ jday.3 : int  6 6 6 6 6 6 6 6 6 6 ...
 $ jday.4 : num  6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 ...
 $ jday2.1: num  1.44 1.44 1.44 1.44 1.44 1.44 1.44 1.44 1.44 1.44 ...
 $ jday2.2: int  16 16 16 16 16 16 16 16 16 16 ...
 $ jday2.3: int  36 36 36 36 36 36 36 36 36 36 ...
 $ jday2.4: num  44.9 44.9 44.9 44.9 44.9 ...
```

The data also include the individual encounter data which are contained in the encounter data file (EDF). Not that, unlike the session specific TDF objects, there is just a single EDF that contains all the encounter information, including a 'session' identifier. Here we look at the structure the EDF:

```
str(rbs.edf)
'data.frame':   494 obs. of  5 variables:
 $ Session   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Individual: Factor w/ 299 levels "BBBBP2B","BBBOP1A",..: 51 288 216 253 295 170 56 249 291 216 ...
 $ Occasion  : int  1 1 1 1 1 1 1 1 1 2 ...
 $ Board     : Factor w/ 50 levels "A1","A10","A2",..: 3 4 7 8 9 12 27 45 48 7 ...
 $ Sex       : Factor w/ 3 levels "F","M","U": 2 2 2 2 2 3 2 2 3 2 ...
```

Now we prepare format the data for analysis using the `data2oscr()` function:

```
rbs.data <- data2oscr(edf = rbs.edf,                        #the EDF
                      sess.col = 1,                         #session column
                      id.col = 2,                           #individual column
                      occ.col = 3,                          #occasion column
                      trap.col = 4,                         #trap column
                      sex.col = 5,                          #sex column (optional)
                      tdf = list(rbs.tdf1, rbs.tdf2,        #list of TDFs
                                 rbs.tdf3, rbs.tdf4),
                      K = c(7,5,6,4),                       #occasion vector
                      ntraps = c(50,50,50,50),              #no. traps vector
                      trapcov.names = c("jday","jday2"), #covariate names

                      tdf.sep = "/",                        #char used separator
                      sex.nacode = "U")                     #unknown sex code
```

The helper function `data2oscr()` actually produces several processed data objects. The one we need for analysis using `oSCR` is the `scrFrame`:

```
names(rbs.data)
[1] "edf"      "y3d"      "sex"      "traplocs" "trapopp"  "trapcovs" "scrFrame"
```

```
rbs.sf <- rbs.data$scrFrame
```

This object contains spatially explicit session-specific data objects for convenient and general analyses:

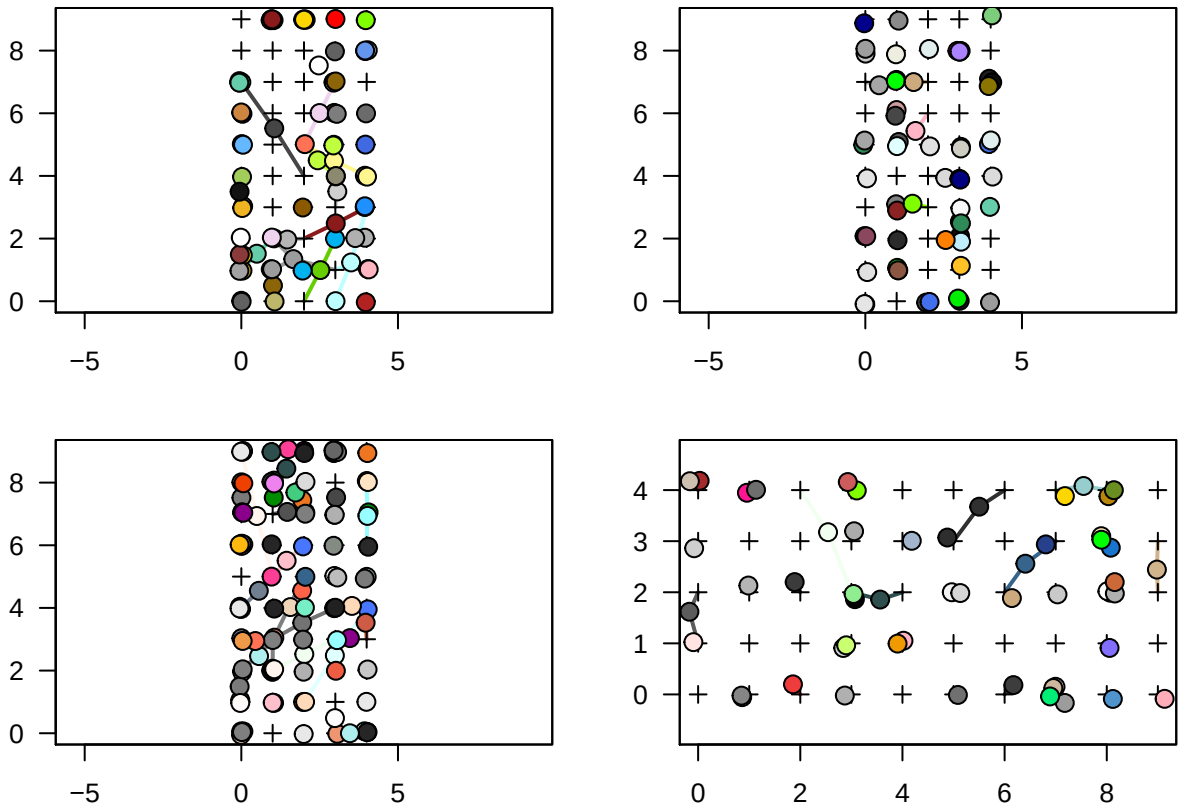
```
names(rbs.sf)
 [1] "caphist"       "traps"         "indCovs"       "trapCovs"      "sigCovs"       "trapOperation" "o
 [8] "type"          "mmdm"          "mdm"           "telemetry"
```

```
?make.scrFrame() #detailed object description
```

A summary of the `scrFrame` can be accessed by simply typing the object name and informative session-specific summaries of the encounter data and monitoring effort are printed:

```
rbs.sf

              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              2.02 1.05 1.68 1.29

Pooled MMDM:  1.66
```

```
par(mfrow=c(2,2), mar=c(2,2,2,2), oma=c(0,0,0,0))
plot(rbs.sf,jit = 2)
```

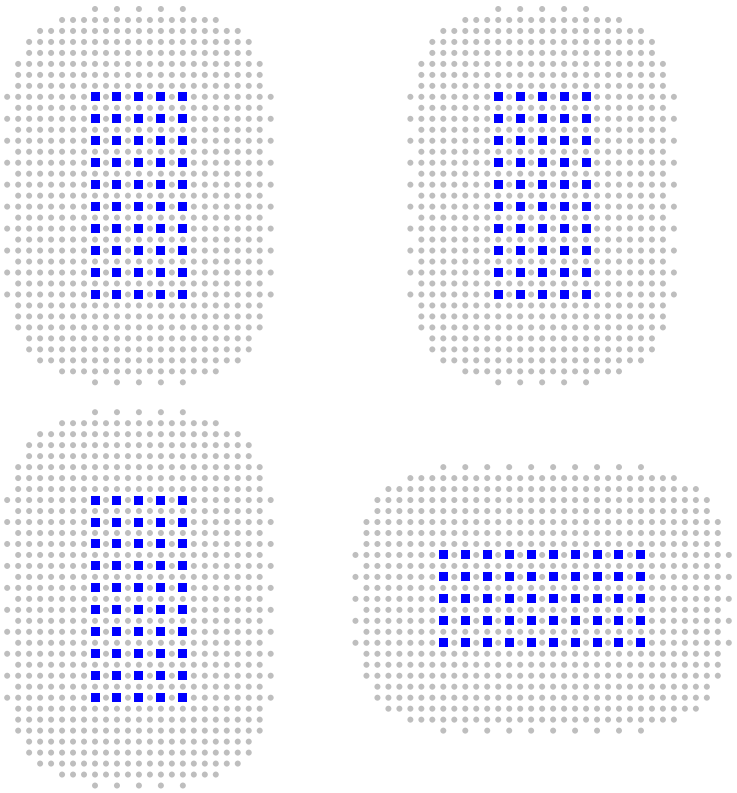


Now we create the state space object using the `make.ssDF()` function.

```
rbs.ss <- make.ssDF(scrFrame = rbs.sf, buffer = 4, res=0.5)
str(rbs.ss)
List of 4
 $ :'data.frame':   757 obs. of  3 variables:
  ..$ X : num [1:757] 0 1 2 3 4 -1.5 -1 -0.5 0 0.5 ...
  ..$ Y : num [1:757] -4 -4 -4 -4 -4 -3.5 -3.5 -3.5 -3.5 -3.5 ...
  ..$ Tr: int [1:757] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "out.attrs")=List of 2
  .. ..$ dim     : int [1:2] 25 35
  .. ..$ dimnames:List of 2
  .. .. ..$ Var1: chr [1:25] "Var1=-4.0" "Var1=-3.5" "Var1=-3.0" "Var1=-2.5" ...
  .. .. ..$ Var2: chr [1:35] "Var2=-4.0" "Var2=-3.5" "Var2=-3.0" "Var2=-2.5" ...
 $ :'data.frame':   757 obs. of  3 variables:
  ..$ X : num [1:757] 0 1 2 3 4 -1.5 -1 -0.5 0 0.5 ...
  ..$ Y : num [1:757] -4 -4 -4 -4 -4 -3.5 -3.5 -3.5 -3.5 -3.5 ...
  ..$ Tr: int [1:757] 2 2 2 2 2 2 2 2 2 2 ...
  ..- attr(*, "out.attrs")=List of 2
  .. ..$ dim     : int [1:2] 25 35
  .. ..$ dimnames:List of 2
  .. .. ..$ Var1: chr [1:25] "Var1=-4.0" "Var1=-3.5" "Var1=-3.0" "Var1=-2.5" ...
  .. .. ..$ Var2: chr [1:35] "Var2=-4.0" "Var2=-3.5" "Var2=-3.0" "Var2=-2.5" ...
 $ :'data.frame':   757 obs. of  3 variables:
  ..$ X : num [1:757] 0 1 2 3 4 -1.5 -1 -0.5 0 0.5 ...
  ..$ Y : num [1:757] -4 -4 -4 -4 -4 -3.5 -3.5 -3.5 -3.5 -3.5 ...
  ..$ Tr: int [1:757] 3 3 3 3 3 3 3 3 3 3 ...
  ..- attr(*, "out.attrs")=List of 2
  .. ..$ dim     : int [1:2] 25 35
  .. ..$ dimnames:List of 2
  .. .. ..$ Var1: chr [1:25] "Var1=-4.0" "Var1=-3.5" "Var1=-3.0" "Var1=-2.5" ...
  .. .. ..$ Var2: chr [1:35] "Var2=-4.0" "Var2=-3.5" "Var2=-3.0" "Var2=-2.5" ...
 $ :'data.frame':   757 obs. of  3 variables:
  ..$ X : num [1:757] 0 1 2 3 4 5 6 7 8 9 ...
  ..$ Y : num [1:757] -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 ...
  ..$ Tr: int [1:757] 4 4 4 4 4 4 4 4 4 4 ...
  ..- attr(*, "out.attrs")=List of 2
  .. ..$ dim     : int [1:2] 35 25
  .. ..$ dimnames:List of 2
  .. .. ..$ Var1: chr [1:35] "Var1=-4.0" "Var1=-3.5" "Var1=-3.0" "Var1=-2.5" ...
  .. .. ..$ Var2: chr [1:25] "Var2=-4.0" "Var2=-3.5" "Var2=-3.0" "Var2=-2.5" ...
 - attr(*, "class")= chr "ssDF"
```

Once created, we can also plot it.

```
par(mfrow=c(2,2), mar=c(0,0,0,0), oma=c(0,0,0,0))
plot(ssDF = rbs.ss, scrFrame = rbs.sf)
```



Now for model fitting. As an example, lets start with the two contrasting models. The *base model* has constant density and space use, and, based on what we know about RBS activity patterns, a detection model with a quadratic effect of day of year and a behavioural response. The *full model* has session-specific density, session- and sex-specific space use, and, in addition to a quadratic effect and behavioural response, detection varying by session and sex.

```
if(1 == 2){ #this takes a *long* time, see below for loading these models
  m1 <- oSCR.fit(model = list(D~1,                  #density
                              p0~b + jday + jday2, #detection
                              sig~1),              #space use
                 scrFrame = rbs.sf, ssDF = rbs.ss)

  m16 <- oSCR.fit(model = list(D~session,                          #density
                               p0~b + jday + jday2 + session + sex, #detection
                               sig~session + sex),                 #space use
                  scrFrame = rbs.sf, ssDF = rbs.ss)
}
```

The full set of models, including all combinations of the models nested within the full model, is as provided below. We continue the workflow with the full set of 16 candidate models:

```
if(1 == 2){ #this takes a *long* time, see below for loading these models
m1 <- oSCR.fit(list(D~1, p0~b + jday + jday2, sig~1),
               rbs.sf, rbs.ss)
m2 <- oSCR.fit(list(D~session, p0~b + jday + jday2, sig~1),
               rbs.sf, rbs.ss)
m3 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + session, sig~1),
               rbs.sf, rbs.ss)
m4 <- oSCR.fit(list(D~1, p0~b + jday + jday2, sig~session),
               rbs.sf, rbs.ss)
m5 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + session, sig~1),
               rbs.sf, rbs.ss)
m6 <- oSCR.fit(list(D~session, p0~b + jday + jday2, sig~session),
               rbs.sf, rbs.ss)
m7 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + session, sig~session),
               rbs.sf, rbs.ss)
m8 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + session, sig~session),
               rbs.sf, rbs.ss)
m9 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + sex, sig~sex),
               rbs.sf, rbs.ss)
m10 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + sex, sig~sex),
                rbs.sf, rbs.ss)
m11 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + session + sex, sig~sex),
                rbs.sf, rbs.ss)
m12 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + sex, sig~session + sex),
                rbs.sf, rbs.ss)
m13 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + session + sex, sig~sex),
                rbs.sf, rbs.ss)
m14 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + sex, sig~session + sex),
                rbs.sf, rbs.ss)
m15 <- oSCR.fit(list(D~1, p0~b + jday + jday2 + session + sex, sig~session + sex),
                rbs.sf, rbs.ss)
m16 <- oSCR.fit(list(D~session, p0~b + jday + jday2 + session + sex, sig~session + sex),
                rbs.sf, rbs.ss)}
```

These models take a while to run, so the fitted models are included with the package:

```
data(rbs_ecography_mods)
ls()[1:16]
 [1] "m1"  "m10" "m11" "m12" "m13" "m14" "m15" "m16" "m2"  "m3"  "m4"  "m5"  "m6"  "m7"  "m8"  "m9"
```

Simply typing the name of a fitted model object produces model coeficient summary tables that are consistent with the format of `unmarked` summary tables. For example::

```
m1
 Model:  D ~ 1 p0 ~ b + jday + jday2 sig ~ 1
 Run time:  202.0398  minutes
 AIC:  3127.07

Summary table:
                Estimate    SE       z P(>|z|)
p0.(Intercept)    -4.398 0.221 -19.896   0.000
t.beta.jday        0.783 0.099   7.871   0.000
t.beta.jday2      -0.128 0.014  -9.229   0.000
p.behav            2.899 0.211  13.735   0.000
sig.(Intercept)   -0.063 0.069  -0.910   0.363
d0.(Intercept)    -0.362 0.110  -3.289   0.001
psi.constant       0.317 0.152   2.090   0.037
*Density intercept is log(individuals per pixel)
  Nhat(state-space) = exp(d0.)*nrow(ssDF)
  (caution is warranted when model contains density covariates)
```

When there are multiple competing models, it is customary to compare models using some metric of quality. In `oSCR` model selection is done using AIC, and is done by creating an `fitList` and applying the model selection function `modSel.oSCR()`:

```
fl <- fitList.oSCR(list(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16),
                   drop.asu=T, rename=TRUE) #rename=T adds sesnible model names
ms <- modSel.oSCR(fl)
```

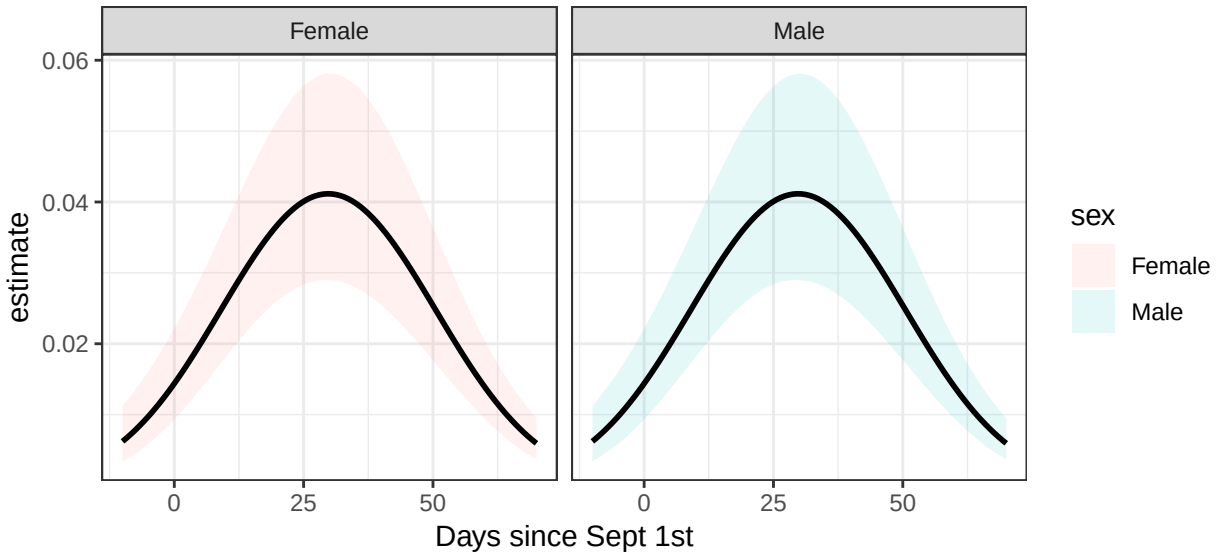
```
ms
 AIC model table:

                                                    model logL  K  AIC  dAIC  weight CumWt
1                     D(~session) p(~b + jday + jday2) sig(~session)  1547 13 3120  0.00 0.31531  0.32
2            D(~session) p(~b + jday + jday2 + session) sig(~session)  1544 16 3120  0.55 0.23936  0.55
3          D(~session) p(~b + jday + jday2 + sex) sig(~session + sex)  1546 15 3122  2.10 0.11037  0.67
4  D(~session) p(~b + jday + jday2 + session + sex) sig(~session + sex)  1543 18 3122  2.53 0.08887  0.75
5                  D(~1) p(~b + jday + jday2 + session) sig(~session)  1548 13 3123  3.07 0.06805  0.82
6                           D(~session) p(~b + jday + jday2) sig(~1)  1552 10 3123  3.28 0.06115  0.88
7                           D(~1) p(~b + jday + jday2) sig(~session)  1552 10 3124  4.54 0.03264  0.92
8       D(~1) p(~b + jday + jday2 + session + sex) sig(~session + sex)  1547 15 3124  4.75 0.02933  0.95
9                   D(~session) p(~b + jday + jday2 + sex) sig(~sex)  1550 12 3125  5.25 0.02279  0.97
10               D(~1) p(~b + jday + jday2 + sex) sig(~session + sex)  1551 12 3126  6.49 0.01227  0.98
11                                 D(~1) p(~b + jday + jday2) sig(~1)  1557  7 3127  7.33 0.00806  0.99
12                D(~session) p(~b + jday + jday2 + session) sig(~1)  1551 13 3128  8.33 0.00491  0.99
13                        D(~1) p(~b + jday + jday2 + sex) sig(~sex)  1556  9 3129  9.28 0.00305  1.00
14          D(~session) p(~b + jday + jday2 + session + sex) sig(~sex)  1550 15 3130 10.41 0.00173  1.00
15                    D(~1) p(~b + jday + jday2 + session) sig(~1)  1555 10 3130 10.70 0.00150  1.00
16               D(~1) p(~b + jday + jday2 + session + sex) sig(~sex)  1554 12 3132 12.43 0.00063  1.00
```

For demonstration puposes, we accept the top ranked model for making predictions. The `get.real()` function is similar to the `predict()` function for `unmarked` (an very widely used `R` package for fitting a suite of models (e.g., occupancy models, N-mixture models, distance sampling) to data collected on unmarked populations), or `glm` model objects. The function takes the model object, an argument identifying which model component to predict from, e.g., density (`dens`), detection ('det'), or sigma ('sig'), and a 'newdata' object that is a data frame used for making predictions. The newdata object should contain columns for all covariates in the model component, including a sex column if sex is included in the scrFrame.

```
library(car)
library(ggplot2)
library(ggthemes)

#pick the model with the lowest AIC
topmod <- fl[[which.min(sapply(fl,function(x) x$AIC))]]

#make a dataframe of values for DETECTION predictions
p.pred.df <- data.frame(jday = rep(seq(-1,7,length=100),2),    #obs range
                        jday2 = rep(seq(-1,7,length=100)^2,2),#obs range^2
                        b=0,                                  #pr(initial capture)
                        sex=rep(c(0,1),each=100))             #binary sex

#now predict
p.preds <- get.real(model = topmod, type = "det", newdata = p.pred.df)
p.preds$sex <- factor(p.preds$sex)
levels(p.preds$sex) <- c("Female","Male")
head(p.preds)
```

	jday	jday2	b	sex	estimate	se	lwr	upr
1	-1.0000000	1.0000000	0	Female	0.006222223	0.001884654	0.003433385	0.01125078
2	-0.9191919	0.8449138	0	Female	0.006719883	0.001979997	0.003768291	0.01195562
3	-0.8383838	0.7028875	0	Female	0.007245610	0.002077120	0.004127007	0.01269078
4	-0.7575758	0.5739210	0	Female	0.007799830	0.002175921	0.004510157	0.01345653
5	-0.6767677	0.4580145	0	Female	0.008382857	0.002276306	0.004918251	0.01425311
6	-0.5959596	0.3551678	0	Female	0.008994886	0.002378196	0.005351674	0.01508065

```
#now plot
ggplot(p.preds, aes(x=jday*10, y=estimate, fill = sex)) +
  geom_ribbon(aes(ymin=lwr,ymax=upr), alpha=0.1, colour=NA) +
  geom_line(size=1) +  facet_grid(.~sex) +
  theme_bw() + scale_color_fivethirtyeight() +
  xlab("Days since Sept 1st")
```

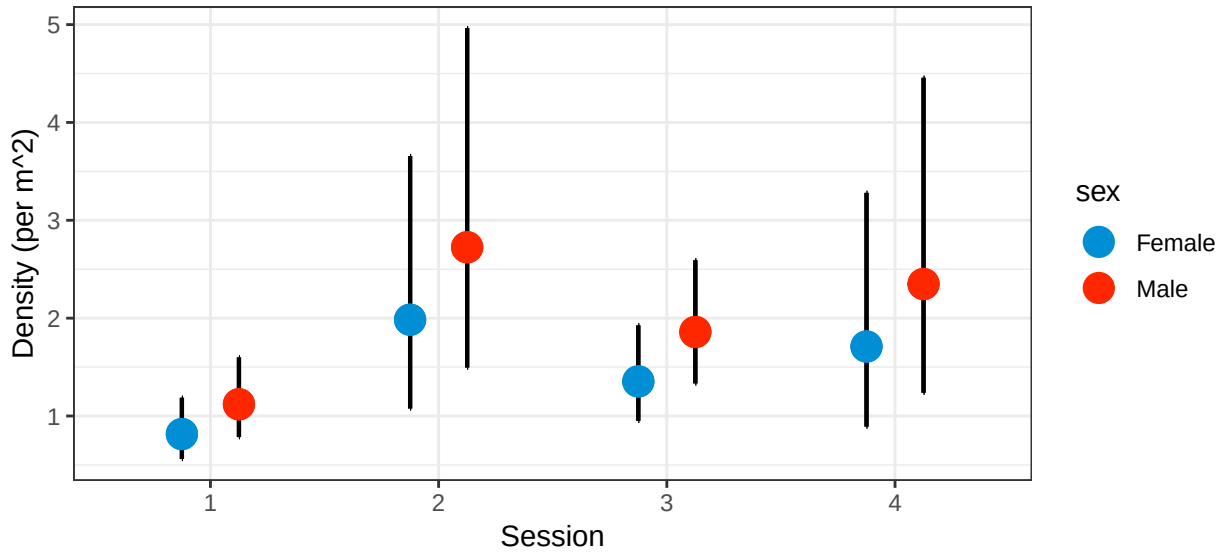


Predictions of density will be on the 'per-pixel' scale. You can change the spatial scale of predictions using the `d.factor` argument. For example, the RBS state space is 0.5 $\times$ 0.5 m, so to report per m^2, include `d.factor = 4`.

```
#make a dataframe of values for DENSITY predictions
d.pred.df <- data.frame(session = rep(factor(1:4),2)) #session specific

#now predict
d.preds <- get.real(model = topmod, type = "dens", newdata = d.pred.df, d.factor = 4)
d.preds$sex <- factor(d.preds$sex)
levels(d.preds$sex) <- c("Female","Male")
head(d.preds)
```

	estimate	se	lwr	upr	session	sex
1	0.8154731	0.1560927	0.5603702	1.186709	1	Female
2	1.9839144	0.6188003	1.0765168	3.656159	2	Female
3	1.3534185	0.2436390	0.9510386	1.926043	3	Female
4	1.7101976	0.5684049	0.8915258	3.280640	4	Female
5	0.8154731	0.1560927	0.5603702	1.186709	1	Female
6	1.9839144	0.6188003	1.0765168	3.656159	2	Female

```
#now plot
ggplot(d.preds, aes(x=session, y=estimate, color = sex, group=sex)) +
  geom_errorbar(aes(ymin=lwr,ymax=upr), width=0, size=0.75, color=1,
                position = position_dodge(width=0.5)) +
  geom_point(size=5, position = position_dodge(width=0.5)) +
  theme_bw() + scale_color_fivethirtyeight() +
  xlab("Session") + ylab("Density (per m^2)")
```



Despite detection not varying between sexes, differences in sample sizes can lead to differences in male vs. female estimates of density.

```
sex.samps <- sapply(rbs.sf$indCovs, function(x) table(x$sex))
sex.samps <- rbind(sex.samps, round(sex.samps[1,] / apply(sex.samps,2,sum),2))
dimnames(sex.samps) <- list(c("F","M","Prop.F"), paste0("S",1:4))
sex.samps
          S1    S2    S3   S4
F      20.00 17.00 27.00 11.0
M      29.00 30.00 33.00 11.0
Prop.F  0.41  0.36  0.45  0.5
```

Finally, lets produce session- and sex-specific values of sigma:

```
#make a dataframe of values for DENSITY predictions
s.pred.df <- data.frame(session = rep(factor(1:4),2),
                        sex = rep(c(0,1),each=4))
#now predict
s.preds <- get.real(model = topmod, type = "sig", newdata = s.pred.df)
s.preds$sex <- factor(s.preds$sex)
levels(s.preds$sex) <- c("Female","Male")
head(s.preds)
   estimate         se       lwr       upr session    sex
1 0.9972654 0.09281836 0.8309704 1.1968397       1 Female
2 0.6387921 0.09726997 0.4739617 0.8609459       2 Female
3 0.9672036 0.08547033 0.8133868 1.1501082       3 Female
4 0.7375126 0.12062608 0.5352369 1.0162319       4 Female
5 0.9972654 0.09281836 0.8309704 1.1968397       1   Male
6 0.6387921 0.09726997 0.4739617 0.8609459       2   Male
```

```
#now plot
ggplot(s.preds, aes(x=session, y=estimate, color = sex, group=sex)) +
  geom_errorbar(aes(ymin=lwr,ymax=upr), width=0, size=0.75, color=1,
                position = position_dodge(width=0.5)) +
  geom_point(size=5, position = position_dodge(width=0.5)) +
  theme_bw() + scale_color_fivethirtyeight() +
  xlab("Session")+ ylab("Sigma (m)")
```