Workflow for an SCR analysis.

Chris Sutherland, David Munoz, David Miller and Evan Grant

Thursday, August 13, 2015

Introduction

In this document, we provide details for a typical workflow for a Spatial Capture-Recapture analysis using the fitting functions provided as supplementary matrials which make up the under-development **R** package `oSCR`. First, we describe the basic data structures that are useful in general SCR analyses. Then, we provide the data and code to reproduce the analysis described in the main paper.

Data structures

The scrFrame Object

The main data object for an SCR anaysis using `oSCR` is the the **scrFrame**, a list of data objects neccessary for fittting an SCR model:

```
scrFrame <- list(caphist,
                 indCovs,
                 traps,
                 trapCovs,
                 trapOperation,
                 type,
                 nocc)
```

The object **scrFrame** is a list containing all of the named objects, which themseleves are lists containing a data object for each of the R session in the analysis. For example, in the analysis in the main paper, we considered four plots, i.e. $R = 4$ sessions.

- `caphist`: A list of 3-dimaensional $n \times J \times K$ arrays, one for each or the R sessions, where n is the number of individuals, J is the number of traps, and K is the number of sampling occasions. The array is filled with 1's or 0's that denote whether individuals were captured (1) or not (0) in each trap in each occasion.
- `indCovs`: A list of data frames, one for each of the R sessions, containing individual level data. Currently, this data oject is only used for *sex* which can be male (1), female (0), or uknown (NA).
- `traps`: A list of data frames, one for each of the R sessions, containing the coordinates of the traps/detectors. The data frame *must* have the headings **X** and **Y**. A data frame for each of R sessions is
- `trapCovs`: A two level list object which is a list of lists of data frames. This list object is of length R, and each of the R slots in the list is itself a list of data frames containing trap level covariates. The objects requires a data frame for each sampling occasion which can be identical of differ by occasion, which allows time varying trap level effects.
- `trapOperation`: A list of binary matrices, one for each of the R sessions, with dimension $J \times K$. The matrix denotes whether a trap was operational (1) or not (0), and if not, the assocaited cells in the the encounter histories (i.e., in **caphist**) do not contiibute to the likelihood.

- `type`: This should be set to **SCR** always. This option will, in the future, allow the user to provide data in the **secr** format.
- `nocc`: An integer denoting the largest number of occasions/visits for any of the R sessions.

While **caphist**, **traps**, **type** and **nocc** *must* be provided, **indCovs**, **trapCovs** and **trapOperation** can be set to 'NULL'.

The State-Space Objects

In addition to the encounter history data and trap information provided in the scrFrame object, a state-space data object is also required for an SCR analysis. Although the fitting function in `oSCR` can generate a state-space object 'on-the-fly', we recommend that users specify the size and resolution of the stat-space in order avoid any errors, but which will also unsure a better understanding of the model structure. We refer to this data object as **ssDF** (the stat-space data frame object).

- `ssDF`: A list of data frames, one for each of the R sessions, containing the coordinates of the state-space pixels used to approximate the landscape of interest. The data frame *must* have the headings **X** and **Y**. These data frames may also have named columns that represent cell-specific (i.e., spacially varying) covariate values.

The Red-Backed Salamnder Analysis

RBS Data

Here we conduct the analysis described in the manuscript demonstrating the data structures outlined above, and how to use the main model fitting function **oSCR**. First you must down load the two files provided as supplement to the paper and place them in a folder somewhere on your computer so you can access them. Now, read in the **R** script that contains all the function neccessary to conduct the analysis (note that here you must specify the full location of where the file has been saved of have set the working directory as such):

```
source("/oSCR.function.script.R")
```

Let's see what we have just loaded here:

```
ls() #list the object in the workspace
```

```
## [1] "e2dist" "oSCR"
```

We have here two functions: "**e2dist**", a function for computing distances between sets of points, and **oSCR**, the integrated likelihood function used to estimate SCR model parameters. The second file contains the salamander data:

```
load("/rbs.scrFrame.Rdata") #the scrFrame
load("/rbs.ssDF.Rdata")     #the ssDF
```

Now, if we check what objects are in the **R** workspace:

```
ls() #list the object in the workspace
```

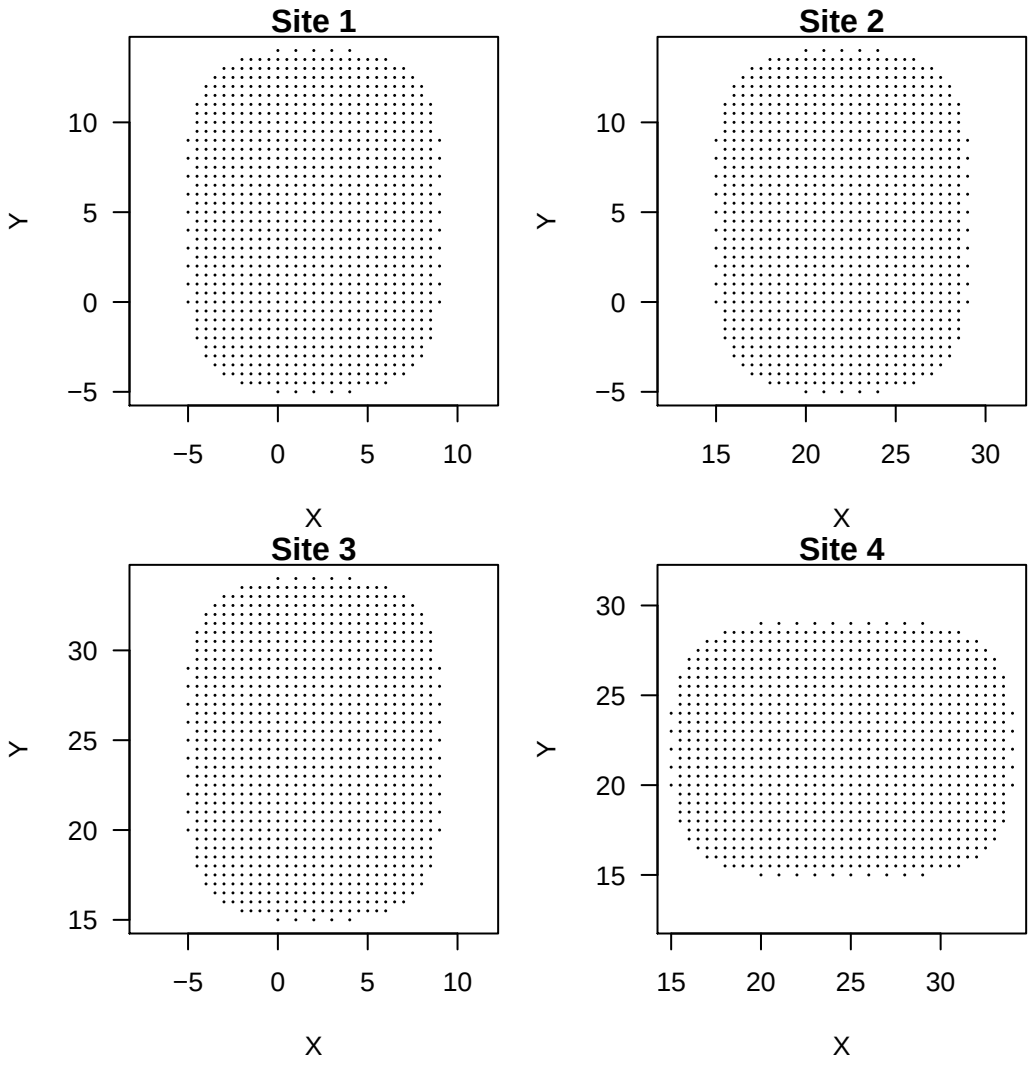
```
## [1] "e2dist"   "oSCR"     "scrFrame" "ssDF"
```

we can see that the **scrFrame** and the **ssDF** objects are also present. To confirm the structure of the data objects, they can be inspected using the `str()` function (although we leave that for you to do):

```
str(scrFrame)
str(ssDF)
```

Instead, we can visualize the data to show the two data objects. In the Red-Backed Salamander example, we analyzed data from four plots, and therefore the various ists should contain data for each plot. For example, there should be a state space for each plot, which of course there is:

```
par(mfrow=c(2,2),oma=c(0,0,0,0),mar=c(4,4,1,1))
plot(ssDF[[1]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 1",las=1) #plot 1
plot(ssDF[[2]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 2",las=1) #plot 2
plot(ssDF[[3]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 3",las=1) #plot 3
plot(ssDF[[4]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 4",las=1) #plot 4
```



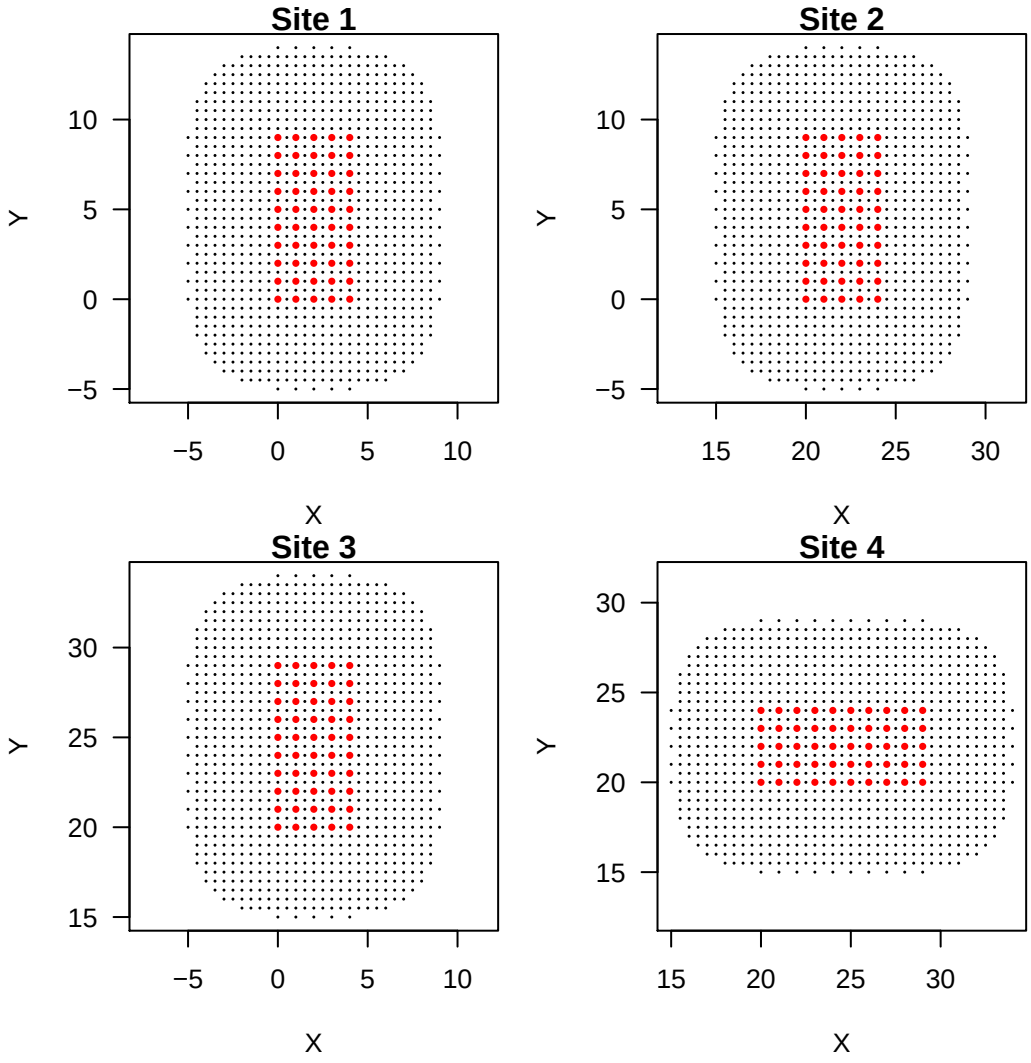
We can also add the trap locations contained in **scrFrame** to the figures (shown as red dots):

```
par(mfrow=c(2,2),oma=c(0,0,0,0),mar=c(4,4,1,1))
plot(ssDF[[1]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 1",las=1) #plot 1
points(scrFrame$traps[[1]][,c("X","Y")],pch=16,cex=0.6,col=2)

plot(ssDF[[2]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 2",las=1) #plot 2
points(scrFrame$traps[[2]][,c("X","Y")],pch=16,cex=0.6,col=2)

plot(ssDF[[3]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 3",las=1) #plot 3
points(scrFrame$traps[[3]][,c("X","Y")],pch=16,cex=0.6,col=2)

plot(ssDF[[4]][,c("X","Y")],pch=16,cex=0.25,asp=1,main="Site 4",las=1) #plot 4
points(scrFrame$traps[[4]][,c("X","Y")],pch=16,cex=0.6,col=2)
```



Model Fitting

Now that we have created the two main data objects, namely **scrFrame** and **ssDF**, we can fit some of the models described in the Red-Backed Salamnder analysis in the main paper. For brevity, we will demonstrate

the fitting of four models using the fitting function `oSCR`, but first we will describe the basic structure and requirements of the function that takes the following arguements:

```
oSCR <-function(
  scrFrame=scrFrame,
  model = list(D~1, p0~1, a1~1, path~1),
  ssDF = NULL,
  trimS = NULL,
  start.vals = NULL,
  ...}
```

- `scrFrame`: The **scrFrame** object described above that contains encounter history data, trap coordinates, and any individual and trap level covariates.
- `model`: This is a list containing four formula arguments that are used exactly like standar linear models in **R**. The first model in the list is the model description for *density* (D ~ .), the second is the model for the baseline detection probability (p0 ~ .), the third is the model for the spatail scale parameter (a1 ~ .), and the foueth and final model is for modeling cost/resistance surfaces (path ~ .). The order of the models matter and therefore care must be taken to ensure the correct ordering of the the list object, which for a null model (i.e., no covariates) look like this: `list(D 1, p0 1, a1 1, path 1)`. We do not discuss the fourth model formula as it is not used in this analysis, and it is set to `path 1` for all models.
- `ssDF`: The state-space object described above that contains the coordinates of the distritized state-space pixel centroids and any associated spatially vrying covariates of interest.
- `trimS`: The is an distance value that restrictes the area of integration to a local area to within `trimS` distance units from any traps an individual is observed at. The idea being that, given a set of spatail locations where an individual has been observed, only locations where that individual is has a non-zero likelihood of having its' activity center are evaluated. Setting this value to around $3 \times \sigma$ can speed up computation without introducing any biases in parameter estimates.
- `start.vals`: A vector of starting values can be provided if required. Again, doing so can speed up computation. The length of the starting value vector must be the same as the number of parameters being estimated.

Now we can fit the models:

```
#these take some time to run:
sal1 <- oSCR(scrFrame,model=list(D~1,       p0~day+day2, a1~1), ssDF=ssDF, trimS=6)
sal2 <- oSCR(scrFrame,model=list(D~session, p0~day+day2, a1~1), ssDF=ssDF, trimS=6)
sal5 <- oSCR(scrFrame,model=list(D~1,       p0~1,        a1~1), ssDF=ssDF, trimS=6)
sal7 <- oSCR(scrFrame,model=list(D~session, p0~1,        a1~1), ssDF=ssDF, trimS=6)
```

Rather than wait for these models to run in order to inspect the output, the output for all the models fitted in the RBS analysis can be loaded and inspected directly:

```
load("/rbs.results.Rdata")
```

Now we can take a look at what the output from a typical model would look like from **oSCR**. We can look at the parameter estimates from model 1 (`sal1`):

```
sal1$outStats
```

```
##    parameters    link    mle  mle.tr se.tr
## 1         p0. (logit) -2.979   0.048    NA
## 2         a1.   (log) -0.593   0.552    NA
## 3 n0.constant   (log)  5.738 310.435    NA
```

or we can compare the AIC values for each of the four models:

```
data.frame(model = paste0("sal",c(1,2,5,7)),
           AIC   = c(sal1$AIC,sal2$AIC,sal5$AIC,sal7$AIC))
```

```
##   model      AIC
## 1  sal1 3482.190
## 2  sal2 3483.107
## 3  sal5 3435.580
## 4  sal7 3478.477
```

For any questions, queries, or comments regarding the content in this workflow document, please feel free to contact me (Chris Sutherland) at chrissuthy@gmail.com. We not that the above analysis can also be coducted using the **R** package `secr`. Documentation for that package, including many worked examples, can be found here: http://www.otago.ac.nz/density/SECRinR.html. In addition, the user groups are an extrememly rich source of information and guidance:

- 'Spatial Capture-Recapture' forum: https://groups.google.com/forum/#!forum/spatialcapturerecapture
- 'secr' forum: https://groups.google.com/forum/#!forum/secrgroup