concordance=TRUE

A tutorial for Step Selection Function

P. Antkowiak[*] H. Tripke[†] C. Wilhelm[‡]

November 25, 2014

Contents

[*]M.Sc. programme ”GIS und Umweltmodellierung” at University of Freiburg
[†]M.Sc. programme ”Wildlife, Biodiversity and Vegetation” at University of Freiburg
[‡]M.Sc. programme ”Wildlife, Biodiversity and Vegetation” at University of Freiburg

1 Introduction

In addition to Resources Selection Functions (RSF) another powerful tool for evaluating data on animal movements and habitat selection are Step Selection Functions (SSF). The latter are used to estimate resource selection by comparing observed habitat use with available structures. Given GPS locations of a collared individual each observation is connected by a linear segment. These segments are considered as steps. The time intervals influencing the step length should be choosen carefully (i.e. by conducting a pilot study) to meet the requirements of the study questions and the target species. The SSF than calculates random steps by taking measured angle and distance along steps and using the observed position as starting points. These alternative positions represent the available habitat beside observed positions. Finally, a comparison of spatial attribute on both describes the habitat selection made be animals [2].
So far, SSF models were mainly done using Geospatial Modelling Environment (GME) that works with a GIS (www.spatialecology.com/gme/). However, more and more packages for analyzing animal movements are provided in R. Non of these packages is designed for doing a SSF only but different ones provide helpful functions that perform single steps of the Selection Function. Therefore, the aim of this tutorial is to collect all functions necessary to conduct a SSF and order them in a way that intuitively makes you understand how to run a SSF with your own data. Each step will be explained using an exemplary dataset of GPS locations collected from seven Cougars (*Puma concolor*) in the year 2010 (in the following adressed as `xmpl`.

Figure 1 provides an overview of necessary steps and potential options to conduct a SSF. The initial data need to be stored in two independant datasets:

1. a raster file of your spatial attributes and
2. GPS locations of your individuals asigned with a time stamp

. Main focus lays on the right site of the workflow, describing preparation of the waypoint data. After loading the table into R you need to create an so-called `ltraj` object. This data class can now be further transformed by ... Random steps shold only be calculated for equal time intervals. These can be defined by creating bursts. Each burst has an unique ID (often including an ID for the individual and the time stamp). While there are many options to adjust your waypoint data the raster data describing your spatial attributes needs not much of imput. Once you created random steps for your observed positions you can extract the spatial attributes for each of those positions by using the fucntion `extract`. At this point waypoint and raster data will be combined and your final model can be written.

In our tutorial we use telemetry data from Cougars/Mountain Lion (*Latin name*) collected by Simone Ciuti[1]. As spatial parameters x tables for ruggedness, slope, canopy cover etc. are available. Describe study area... DO WE NEED A EQUATION??

$$w(x) = exp(\beta 1x1 + \beta 2x2 + ... + \beta pxp)$$

Why did we not use the data (Wildboar) prepared for the adehabitatLT package? - No information on details, metadata provided, it is hard to understand when to use which dataset and why.

1.1 Preparations

Before you can actually start using the tutorial for conducting SSF you need to load a bunch of packages in R. Some of them require others so that you have to add all these to your library:

[1]we might have to specify that and name and thank the institute.../collected/containing

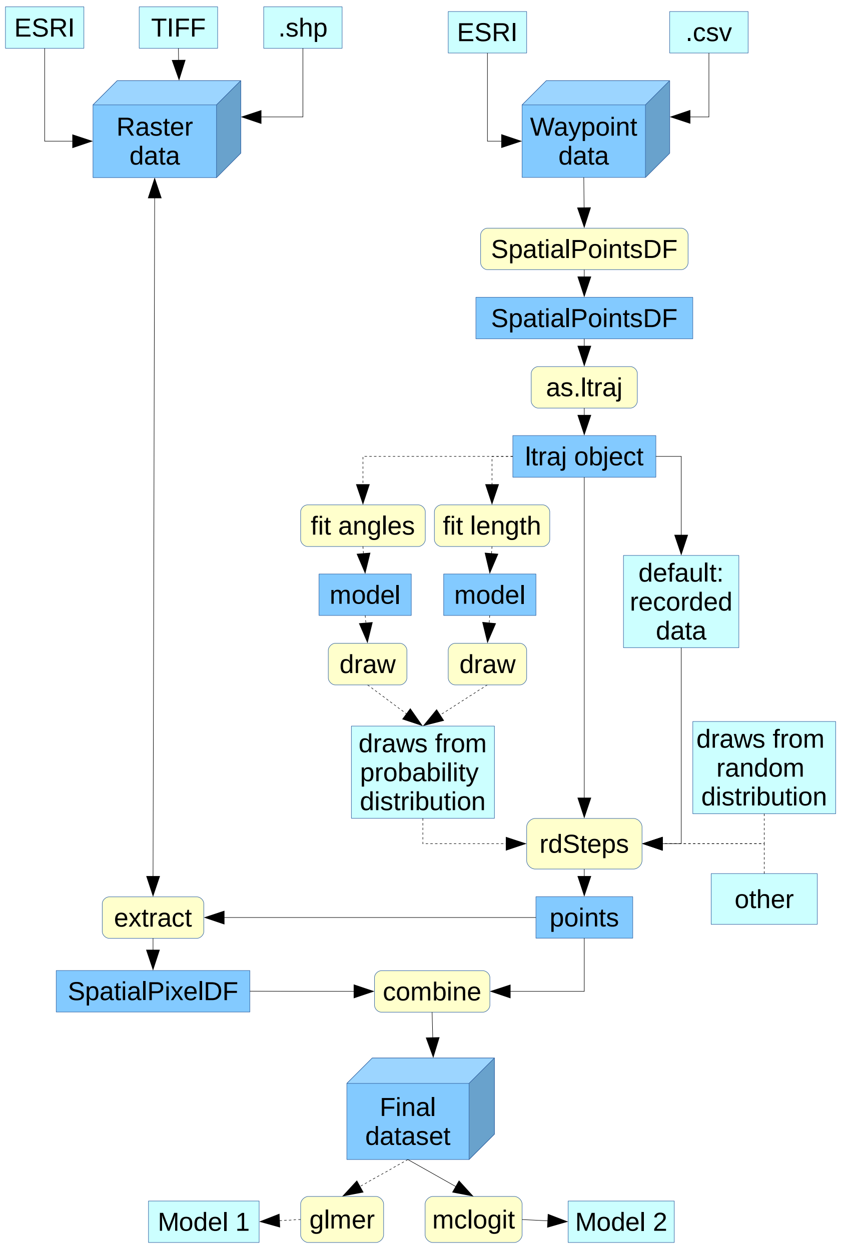

Figure 1: Conducting a Step Selection Function using existing R-packages. The yellow boxes show the name of the fucntion applied while the blue boxes provide the type of object or data. Following the arrows a step by step instruction is provided ...

1.2 Packages - what we need

```
# installing packages ------------------------------------------------------
## for implementing SSF
# install.packages("adehabitat") # outdated version, not needed for this tutorial
install.packages("adehabitatHR")
install.packages("adehabitatHS")
install.packages("adehabitatLT")
install.packages("adehabitatMA")
install.packages("tkrplot")
install.packages("hab", repos = "http://ase-research.org/R/") # regular
install.packages("hab", repos = "http://ase-research.org/R/", type = "source") # for s

# for handling ratser data
install.packages("move")
install.packages("raster")
install.packages("rgdal")
#install.packages("")

# loading the packages
# require(adehabitat) # keep fingers off this package. It is outdated.
require(hab)
require(adehabitatMA)
require(adehabitatHR)
require(adehabitatHS)
require(adehabitatLT)


## for i dont know

#require(move)
#require(raster)
#require(rgdal)
#require(tkrplot)
#require(raster)
#require(sp)
```

2 All right site steps

2.1 Load telemetry data (*.csv, ESRI)

The data for the analysis should be safed in a simple *.csv format. Depending on your analysis you have to include

1. coordinates
2. ID
3. date and/or time
4. ...

2.2 Create a Spatial Points Data Frame

2.3 Create a ltraj object

2.4 Compute random steps

The function **rdSteps** removes the first and the last data point. That's what you want.

3 Spatial covariates

This section explains the use of spatial parameters that will be tested for selection by the target species. You should store these data in raster files (probably ESRI (*.adf) or *.tif) and for time reasons already clipped to your area. How you can do this in R please read the GIS instructions from the other group ;)

3.1 Load raster data (ESRI, *.tif, (*.shp))

With a simple function stored in the package **raster** you are able to upload any ratser file into R. Examplarily we are using the ratser data for ruggedness and canopy cover for the study area.

```
#install.packages("RArcInfo")
#require(RArcInfo)
require(raster)
require(rgdal)

require(sp)



#?raster
#getwd()
#setwd("/home/Peter/")

ruggedness <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW GI
# plot(ruggedness) # outcomment this if you just quickly want to run the script. Takes

landcover <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW GIS
# plot(landcover) # outcomment this if you just quickly want to run the script. Takes a

canopycover <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW G
# plot(canopycover) # outcomment this if you just quickly want to run the script. Takes

disthighway <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW G
# plot(disthighway) # outcomment this if you just quickly want to run the script. Takes

distroad <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW GIS
plot(distroad) # outcomment this if you just quickly want to run the script. Takes a m
```

3.2 Extract coordinates for comparison of used and random points

Peter is successfully doing this step!!

¡¡¡¡¡¡¡ HEAD

4 Load telemetry data (*.csv, ESRI)

The data for the analysis should be safed in a simple *.csv format. Depending on your analysis you have to include

1. ID
2. coordinates
3. date and/or time
4. ...

5 Create a Spatial Points Data Frame

6 Create a ltraj object

7 Creating Bursts

For analysing the data, there might be the need to create "sub-bursts" within your trajectory. For example, if the individuals were only recorded during the day, the monitoring took place over two consecutive years or the time lag between the relocations differs remarkably. Looking at those different parts seperately might be necessary for different reasons. The function `cutltraj` splits the given bursts of your `ltraj` object into smaller burst according to a specified criterion. In contrast, the function `bindltraj` combines the bursts of an object of class `ltraj` with the same attribute "id" to one unique burst. [1] To find out if there are more missing values, you can plot the `ltraj` object. For that, you need to define the time interval you are looking at.

In our example, the locations of the cougars were recorded every 3 hours, starting at 3 AM. The location at midnight is always missing. We now want to split the existing bursts (individuals) into "sub-bursts" where the time lag is smaller than 3 hours. To get an impression about the time lags we plotted the different bursts (individuals).

```r
plotltr(xmpl.ltr, "dt/3600")

# time log 1 hour

plotltr(xmpl.ltr, "dt/3600/3")

# time log 3 hours
```

Because we want to keep relocations which are only a few minutes "wrong", we need a function which defines `dt`, which is the time interval between successive relocations (measured in seconds).

```r
foo = function(dt) {return(dt> (3800*3))}
```

Then we split the object of class `ltraj` according to that function into smaller bursts. The bursts we had before applying this function still remain.

```r
xmpl.cut <- cutltraj(xmpl.ltr, "foo(dt)", nextr = TRUE)
```

8 Compute random steps

The function **rdSteps** removes the first and the last data point. That's what you want.
======= ¿¿¿¿¿¿¿ fbdcfe621d003f329a5c7d1fe40dce142ac0c74d

9 Final SSF model

10 Acknowledgements

Don't forget to thank TeX and R and other opensource communities if you use their products! The correct way to cite R is shown when typing "`citation()`", and "`citation("mgcv")`" for packages.

References

[1] Clement Calenge. Analysis of animal movements in r. 2011.

[2] Henrik Thurfjell, Simone Ciuti, and Mark S Boyce. Applications of step-selection functions in ecology and conservation. *Movement Ecology*, 2(1):4, 2014.