concordance=TRUE

A tutorial for Step Selection Function

P. Antkowiak* H. Tripke† C. Wilhelm‡

November 24, 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) a more detailed anlysis for telemetry data can be conducted by using Step Selection Function (SSF). The use of latter is providing answers to th actual selection of animals on their habitat rather than analysing the use of a habitat only. (DO WE WANT TO CITE, HERE FOR EXAMPLE Simone and friends). So far most of the SSF were done in GIS to use spatial data together with mathematics equations. However, more and more packages are provided in R and thus becomes a valuable alternative. This tutorial provides an overview on how to implement SSF with R. The main package we will use in the following is the package **adehabitatLT**. All single steps that need to be taken care of are summarized in (Figure 1). 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... 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.

2 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:

2.1 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# install.packages("adehabitat") # outdated version, not needed for this tutorial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
```

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

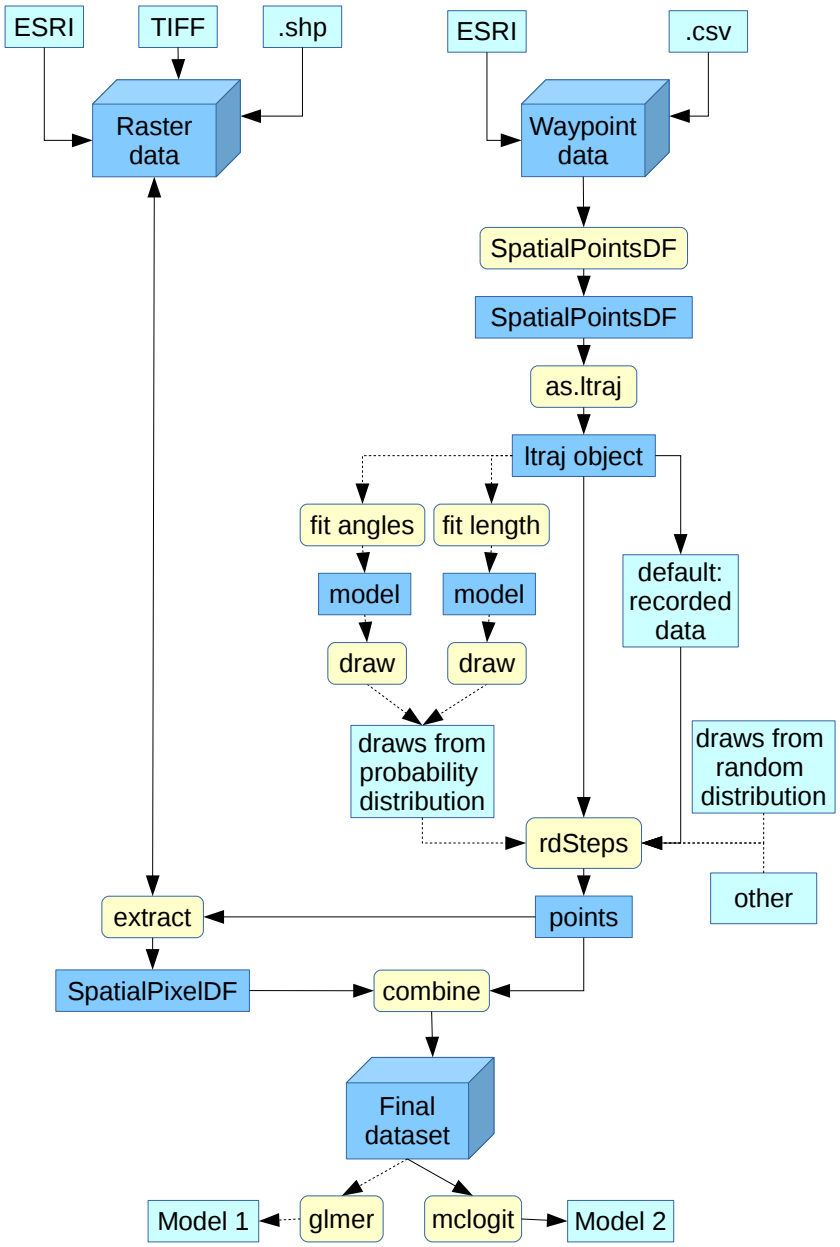

Figure 1: *Conducting a SSF using existing R-packages:* this figures provides an overview of the steps necessary to conduct a SSF. The steps are separated in subsections, which the turtorial will guide you through.

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

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

canopycover <- raster("/home/Peter/Dokumente/uni/WS_14_15/Best Practice R/Dataset/NEW
# 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
# 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!!

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. (c. Calenge) 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).

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

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).

```
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.

```
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.

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.