concordance=TRUE

A tutorial for Step Selection Function

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

November 19, 2014

Contents

1 Introduction

This tutorial is for all those who would like to use Step Selection function (SSF) for their telemtry data...

2 Praparations

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:

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

Every method should contain a figure of study organism, site or yourself in front of some picturesque landscape (see Fig. **??**).

[*] 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

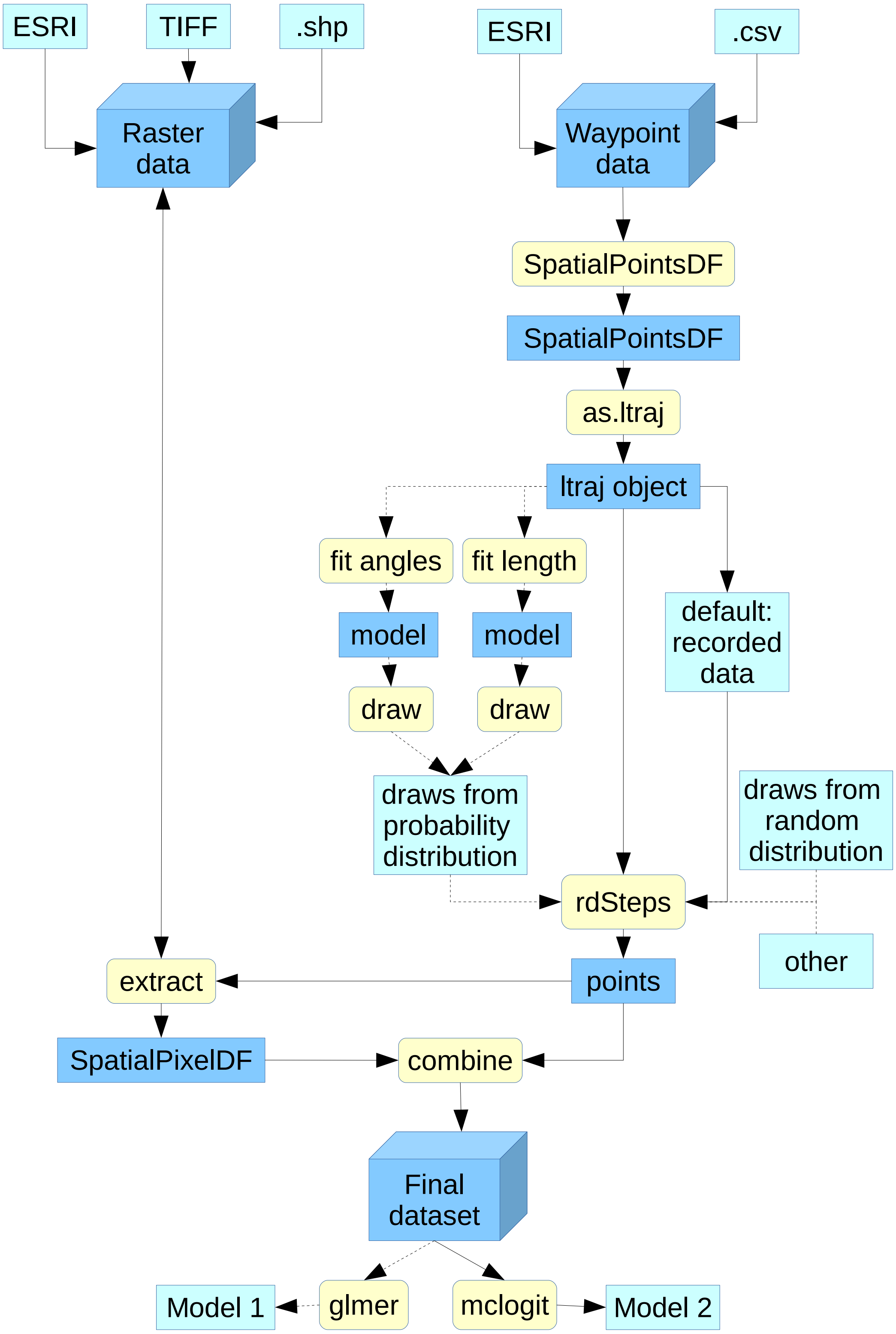


Figure 1: *Conducting a SSF using existing R-packages:* this figures provides an overview of the necessary steps necessary to conduct a SSF. The steps are separated in subsections, which the turtorial explains in detail. `http://en.wikipedia.org/wiki/Template:POTD/2014-08-27`

```
# installing packages -----------------------------------------------------

# install.packages("adehabitat")

# install.packages("hab")
install.packages("hab", repos = "http://ase-research.org/R/", type = "source")
# install.packages("hab", repos = "http://ase-research.org/R/")

install.packages("adehabitatMA")
install.packages("adehabitatHR")
install.packages("adehabitatHS")
install.packages("adehabitatLT") # will be installed when insatlling adehabitatHR
install.packages("tkrplot")


require(hab)
require(adehabitatMA)
require(adehabitatLT)  # includes "ade4"
require(adehabitatHS)
require(tkrplot)

# require(adehabitat)  # not necessary to load
```

4 Extract coordinates for comparison of used and random points

5 Load telemetry data (*.csv, ESRI)

6 Create a Spatial Points Data Frame

7 Create a ltraj object

8 Compute random steps

9 Final SSF model

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.

10 still useful code from Carsten

```
# This is how I could document code:
runif(100)
# but it only contains the code and can also run over the line, as shown here ...
# convenient for short, but not for long pieces of code
# any symbol is plotted "as is", even if it is \LaTeX: *+#&ß\/
```

any symbol is plotted ”as is”, even if it is LaTeX: *+ß

Alternatively, in Sweave, I can actually use R-code and have it evaluated by R and results returned and pasted into the LaTeX-document! Like so:

```
runif(10)

##  [1] 0.4907906 0.5595449 0.3118619 0.6093503 0.6367003 0.4866672 0.4215213 0.4789160
##  [9] 0.8654661 0.1029930
```

Or I can only have the code returned, but **not** evaluated (useful if it takes a long time, or if results are produced externally beforehand):

```
runif(10)
```

Similarly, I can have the code evaluated but the call not returned:

```
##  [1] 0.1721317 0.3118692 0.5463543 0.6010523 0.2735575 0.5221804 0.1843534 0.6345959
##  [9] 0.3507921 0.9575469
```

To combine both, simply add both options, separated by a “,”.

You can also put an R-expression into the text. For example, the mean of 10 random number between 0 and 1 is (in this simulation) 0.541, rounded to 3 digits. This value will change every time you compile this document.

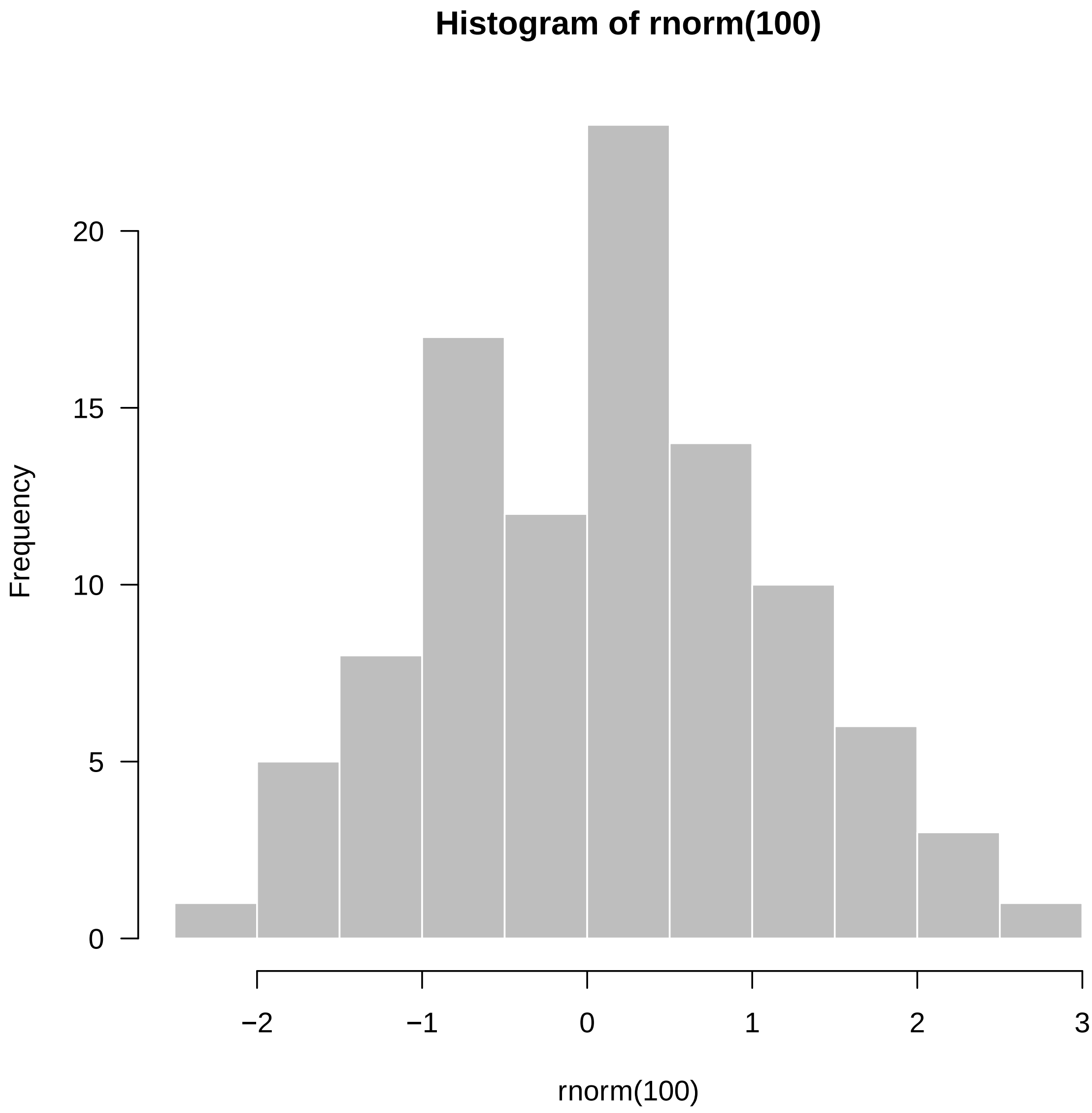
Finally, you can first do the computations, store them (in R objects) and then refer to them in the text.

```
##
##  Pearson's product-moment correlation
##
## data:  X and Y
## t = 0.1449, df = 8, p-value = 0.8884
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.5977195  0.6595417
## sample estimates:
##        cor
## 0.05116057
```

So, the correlation between X and Y is not significant, and the mean of Y is 0.615.

So that’s all cool, and the only thing we also could wish for is to produce figures and insert them into the text automatically:

```
hist(rnorm(100), col="grey", border="white", las=1)
```



We may want to “hide” the figure generating text and show the result in a float-environment with a proper caption. To do so, we only have to wrap the above code in a figure environment, like so:

```
\begin{figure}
\centering
#<<label=fig2plot, echo=FALSE, fig=TRUE>>=
hist(rnorm(100), col="grey", border="white", las=1)
#@
\caption{This is an example figure, computed and immediately returned.}
\end{figure}
```

Notice that I only put the # in to prevent R from evaluating the code. In a real document, you would NOT do that.

Now one minor issue is that you cannot readily change the size of this figure. The best way to do this is to split the figure-generating process up into two part: one generates the figure and saves it as PDF, then we use standard LaTeXto load this figure. Like so:[1]

```
#<<anexamplehistogram, echo=FALSE, dev="pdf">>=
```

[1] Obviously in the document you would have to remove the #s.

Figure 2: This is an example figure, exported as PDF and then, on loading, scaled to half the text's width.

```
hist(rnorm(100), col="cornflowerblue", border="white", las=1)
#@

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{anexamplehistogram}
\caption{This is an example figure, exported as PDF and then, on loading,
  scaled to half the text's width.}
\end{figure}
```

I leave it to you to do the same thing for tables, using R's xtable command, as illustrated in the Sweave demo of Friedrich Leisch (`users.stat.umn.edu/~geyer/Sweave/foo.pdf`).

All this was Sweave. More recently, knitr made its entrance, and has increased the flexibility of Sweave. It is VERY similar and you have to tell RStudio in the options, whether you are sweaving or knitting. For minimal examples and comparisons, also using markdown rather than LaTeXand PDF or html as output, see here: `http://yihui.name/knitr/demo/minimal/`.