Vignette wildxing

Guillaume Bastille-Rousseau

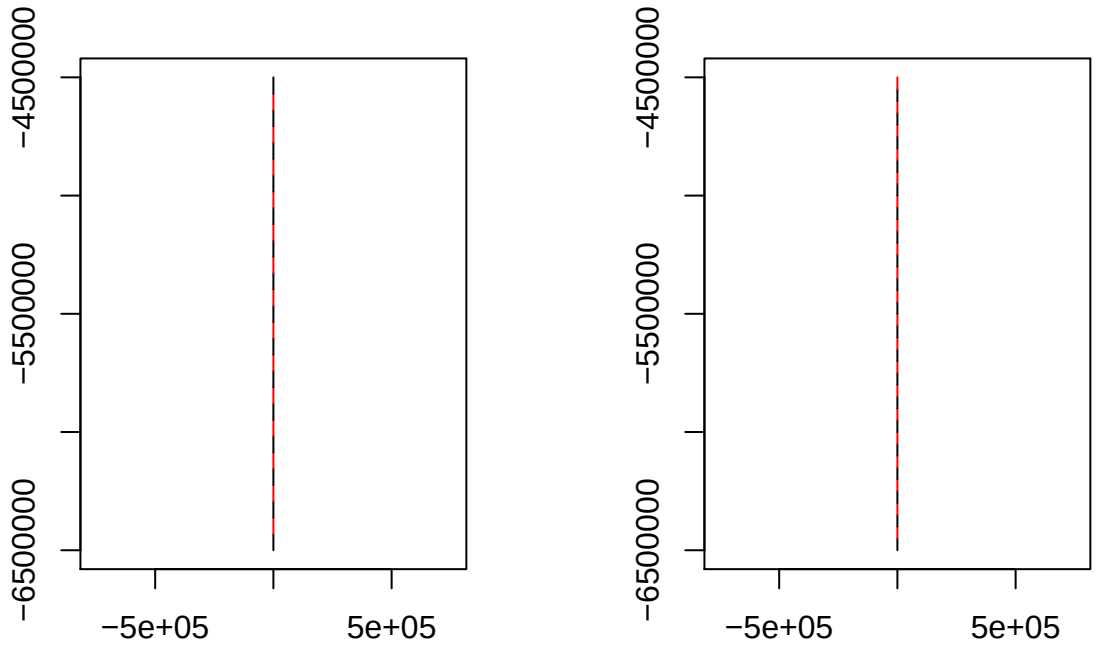
August 2, 2017

The overall approach relies on a few simple steps: 1- Segmenting the linear features into segment of equal length, 2- Intersecting an animal trajectory with the segmented corridor, 3- Repeating for all individuals, 4- Optimizing based on population level responses. An additional function is also available that intersect home-range of an individual with the linear corridor to evaluate size of created fragments.

1- Segmenting of linear corridors in smaller fragments - *SegmentSpL*

The function *SegmentSpL* was taken from http://rstudio-pubs-static.s3.amazonaws.com/10685_1f7266d60db7432486517a111c76ac8b.html. This function split a *SpatialLines* object into segments of equal length.The user can specify how the last segment is considered (separated or merged). We can generate a SpatialLines object from scratch, apply the function using different segment lengths, and plot the resulting segmentation:

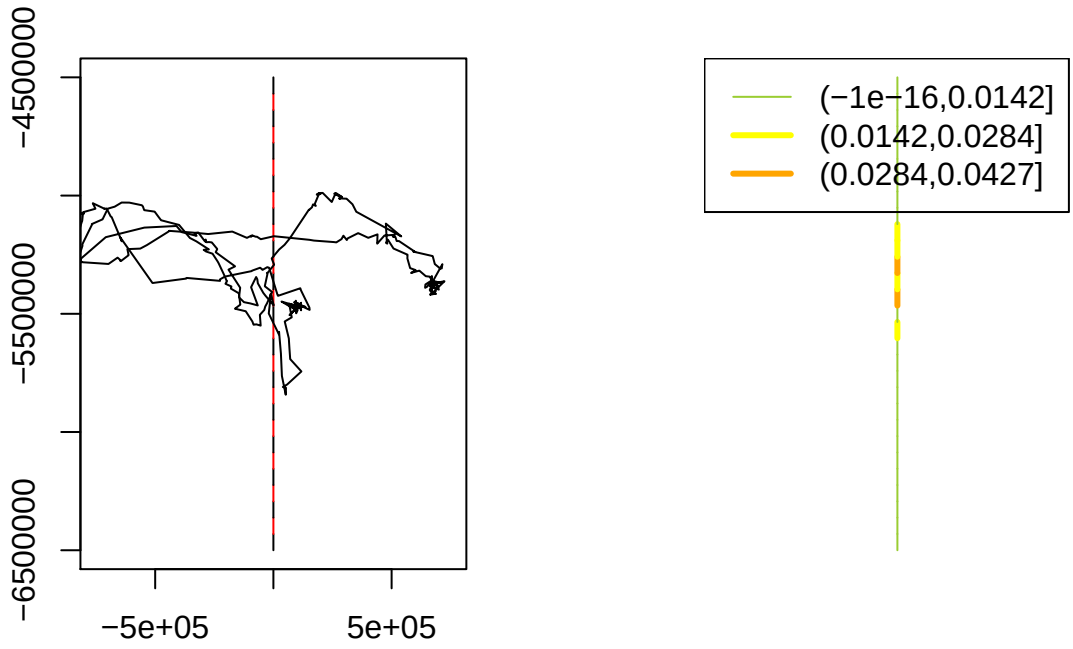
```
x <- c(0,0)
y <- c(-6500000,-4500000)
Spl<-SpatialLines(list(Lines(Line(cbind(x,y)), ID="a")))
t1<-SegmentSpL(Spl, n.parts=30, merge.last=F) #Segment in 20 parts
t2<-SegmentSpL(Spl, length=50000, merge.last=F) #Segment based on length
par(mfrow=c(1,2))
plot(t1, col = rep(c(1, 2), length.out = length(t2)), axes = T)
plot(t2, col = rep(c(1, 2), length.out = length(t2)), axes = T)
```



2- Intersect of an animal trajectory with a corridor - *corriIntersects*

The function *corriIntersects* intersect a trajectory object from *adehabitatLT* with a segmented corridor (the output of *SegmentSpL*). The user can specify if standardisation for the individual occured over the time or number of locations of sampling. THe former is more suitable when individuals have different frequency of location, but long gap in the data should be identified. The function include a plot option and also a separate function *plotcorri_ind*. We will intersect the corridor created earlier with an albatross individual which movement is provided by package *adehabitatLT*. Albatross (and all birds really,) are well known to be impacted by linear features!

```
data (albatross) #From package adehabitatLT
#We are using the segmented corridor from previous step
t3<-corriIntersects(albatross[3], t1, plot=F)
par(mfrow=c(1,2))
plot(t1, col = rep(c(1, 2), length.out = length(t2)), axes = T)
plot(ltraj2sldf(albatross[3]), add=T)
plotcorri_ind(t3, nb_breaks=4)
```



3- Intersect of multiple animal trajectories with a corridor and population average - *corriIntersects_All* and *avg_inds*

wildxing also provides functions to facilitate the execution of the analysis over multiple individuals. *corriIntersects_All* is a wrapper that applies the function *corriIntersects* to all individual (not burst) of a *traj* object and produces a list contianing the analysis for each individual. *avg_inds* then takes this list and compute summary statistics at the population level. We will now perform the analysis using the same corridor as define in step 1, but applying it over all albatrosses.*plotcorri_grp* plot the output of *avg_inds*

```
data(albatross)
t4<-corriIntersects_All(albatross, t1)
length(t4) #6 different individuals
```

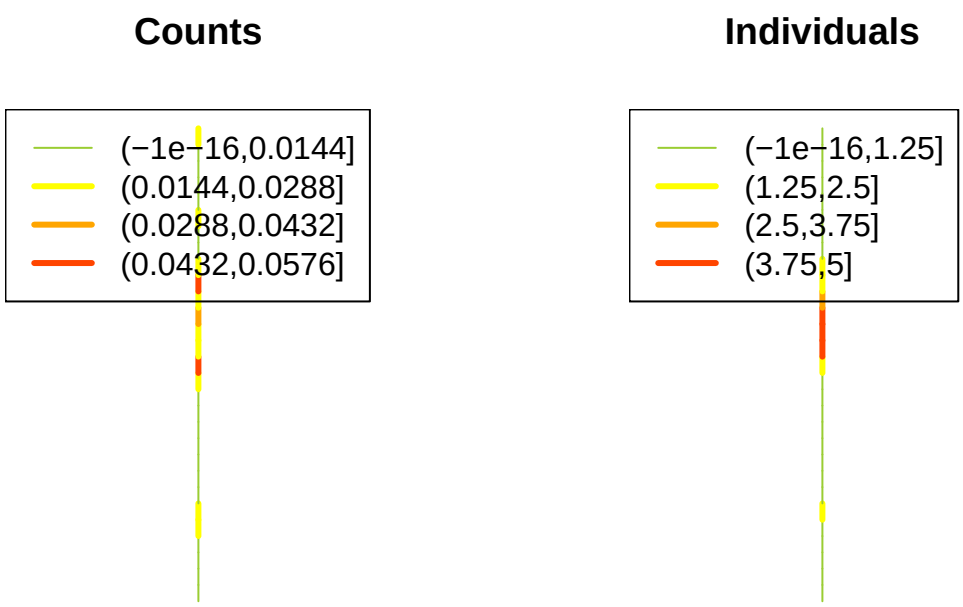
```
## [1] 6
```

```
t5<-avg_inds(t4)
head(t5@data) #View at the column stored
```

```
##   count_mean count_sum  pct_mean_0   pct_mean nb_ind    id
## 1  0.0000000         0 0.000000000 0.00000000      0  None
## 2  0.0000000         0 0.000000000 0.00000000      0  None
## 3  0.1666667         1 0.002207683 0.01324610      1     1
## 4  0.0000000         0 0.000000000 0.00000000      0  None
## 5  0.1666667         1 0.003564076 0.02138446      1     2
```

```
## 6  0.5000000         3 0.009335834 0.02800750      2 Multi
```

```
par(mfrow=c(1:2))
plotcorri_grp(t5, nb_breaks=5, var=4, main="Counts")
plotcorri_grp(t5, nb_breaks=5, var=5, main="Individuals")
```

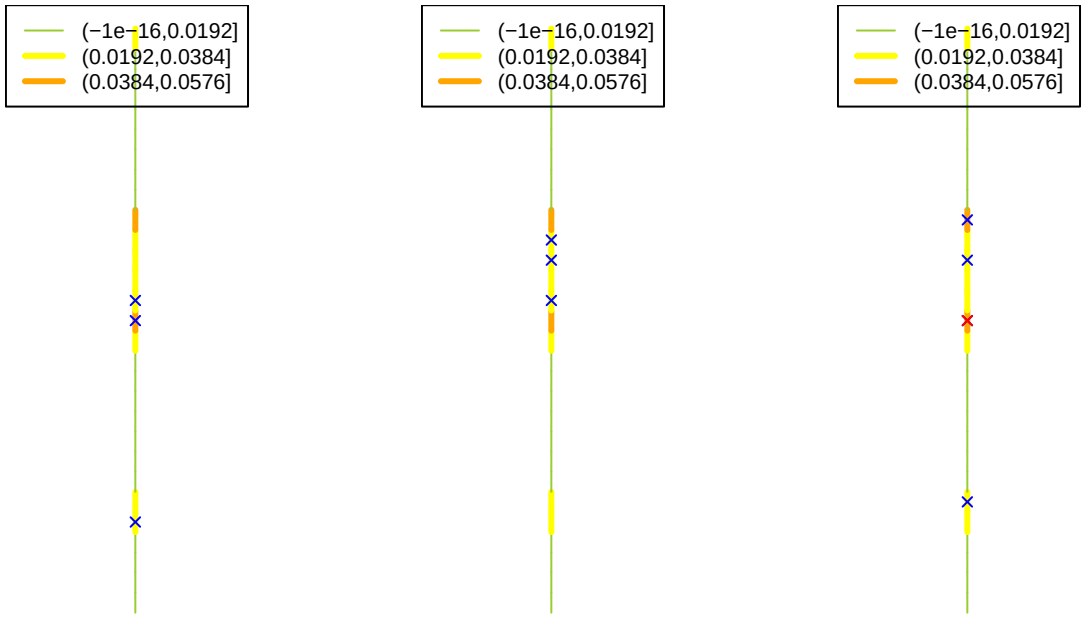


4- Optimization of crossing structures positioning - *corri_optim*

optim_corri takes the output of *avg_inds* and performs an optimization to select ideal segment for wildlife crossing structures. The algorithm maximizes the importance of a segment for crossing and the spatial spread among selected segment. The user can also specify a series of locations where segment that should be excluded or a series of locations where crossing structures will be added by default. These locations should be specified as *SpatialPoints* objects. Many additional arguments can be specifed (see help file for more details). The algorithm called the *Rsymphony* package using the *symphony* solver for linear programming. *plot_optim* plots the output of the function.

```
 #Equal weight, minimum of 1 individual
 opti1<-optim_mclp(t4, var=4, n=3, nb_ind=1, weight=0.5, plot=F)
#More weight to crossing, minimum of 2 individuals
 opti2<-optim_corri(t5, var=4, n=3, nb_ind=3, weight=0.25, plot=F)
 #Equal weight,  additional point
Pts<-SpatialPoints(matrix(c(0,-5500000), nrow=1, ncol=2))
 opti3<-optim_corri(t5, var=4, n=3, nb_ind=1, weight=0.5, add=Pts, plot=F)
  par(mfrow=c(1,3))
```

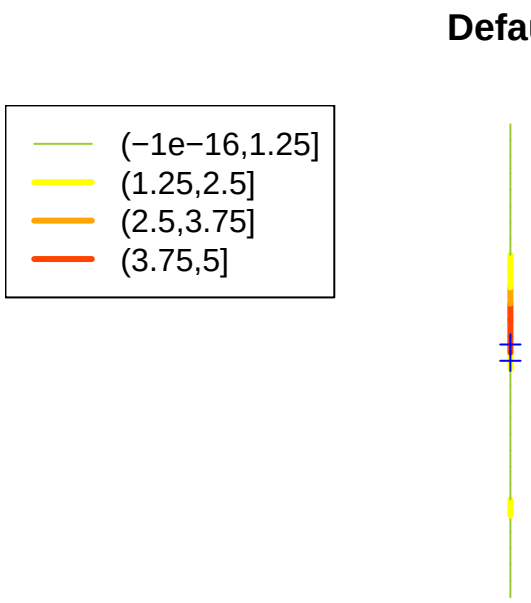
```
plot_optim(t5,  var=4, opti1, main="")
plot_optim(t5,  var=4, opti2, main="")
plot_optim(t5,  var=4, opti3, main="")  #Additional points in red
```



5- Optimization of crossing structures positioning - *corri_mclp*

optim_mclp takes the output of *corriIntersects_All* and performs an optimization to select ideal segment for wildlife crossing structures using the maximum coverage location problem. The user can also specify a series of locations where segment that should be excluded or a series of locations where crossing structures will be added by default. These locations should be specified as *SpatialPoints* objects. Many additional arguments can be specifed (see help file for more details). The algorithm called the *Rsymphony* package using the *symphony* solver for linear programming.

```
#Two crossings
 opti1<-optim_mclp(t4, n=2, dist=5*1000, plot=T)
```



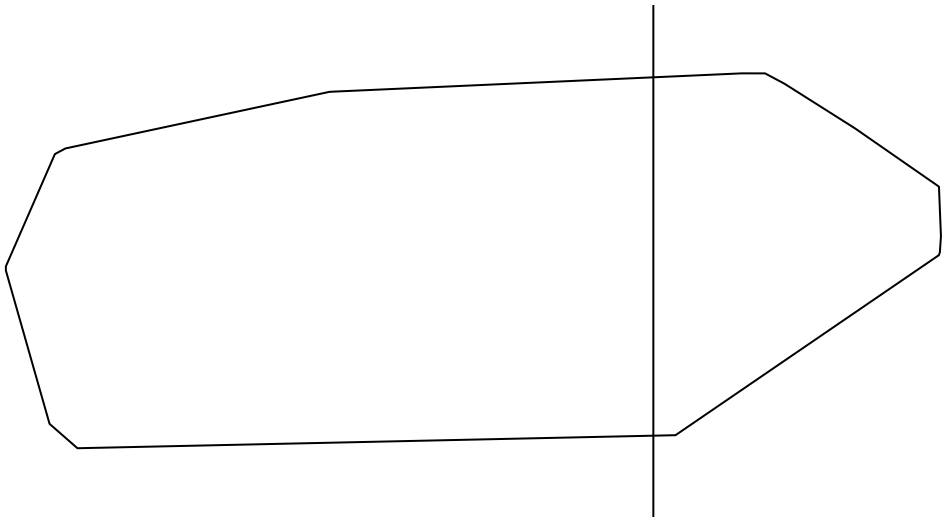

```
opti1[[1]]$objval #All 6 individuals are covered by selected crossing.
```

```
## [1] 6
```

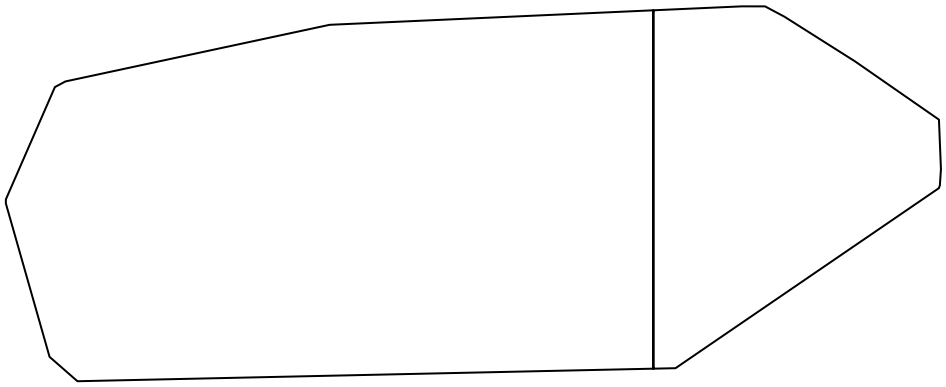
6- Intersect of animal home-ranges with linear features - *hr_split*

We also provide a function that intersects a *SpatialPolygons* object with a *SpatialLines* object and return the segmented polygons and summary statistics. The function *hr_split* will return a list with the first element being the *SpatialPolygons* object. FOr this example, we will keep the same corridor and will generate an home-range for one albatross using the mcp function from the package *adehabitatLT*.

```
 hr<-adehabitatHR::mcp(SpatialPoints(ld(albatross[3])[,1:2]))
plot(hr)
plot(Spl, add=T)
```



```
hr2<-hr_split(hr, Spl)
plot(hr2[[1]])#Plot fragmented home-range
```



```
hr2[2:4] #Summary statistics
```

```
## $`% of bigger fragment`
## [1] 75.06074
##
## $`Nb of fragment`
## [1] 2
##
## $`Area of each fragment`
## [1]  389939.7 1173617.7
```