Home range estimation

Glen A. Sargeant
USGS Northern Prairie Wildlife Research Center
gsargeant@usgs.gov

December 3, 2010

Package wild1 includes `hr.mcp`, a function for computing minimum convex polygon home ranges. Although other routines are available for home range calculation with R, this one integrates classes and functions for spatial data to provide features not found elsewhere. These include 1) clipping of home range boundaries to coincide with study area boundaries, 2) restriction of the domain for data, and 3) use of a robust measure of location for identification of "outliers." This vignette describes `hr.mcp` and presents a series of examples.

Software

Package wild1 and required supporting packages must be installed prior to use. To install wild1, copy the file `wild1.zip` (available at http://www.npwrc.usgs.gov/staff/sargeant.htm) to your preferred location, initiate an R session, and use the pull-down "Packages" menu (*Packages>Install package(s) from local zip files*).

See the package description for a current list of required supporting packages (available from CRAN, the Comprehensive R Archive Network, via the pull-down "Packages" menu). The package description is included with the package documentation (accessible via the pull-down "Help" menu) and can also be viewed with `packageDescription("wild1)`.

Installed packages must be loaded at the start of each R session or they will not be available for use. Either `require()` or `library()` may be used to load wild1. Loading wild1 will automatically load supporting packages.

```
> require(wild1)
```

Minimum convex polygons

Examples that follow rely on 1) a data frame containing coordinates of locations used by elk and on 2) a spatial object that describes the boundary of Wind Cave National Park, and 3) objects of class "gpc.poly"{gpclib} that describe the park boundary and a "hole" (polygonal region to be excluded from the park).

Loads the data and 3 maps:

```
> data(xy.elk)
> data(wica.gpc)
> data(hole.gpc)
> data(wica.hole.gpc)
```

Locations can be described by 2 coordinate vectors or, as in this case, by a matrix. At present, coordinates must describe locations relative to a square grid (e.g., UTM grid).

```
> str(xy.elk,strict.width="cut")
'data.frame':        2444 obs. of  2 variables:
 $ x: num  619347 NA 618902 619727 619193 ...
 $ y: num  4819457 NA 4819067 4819281 4819279 ...
```

Any study area defined by a single polygon may be described by a matrix, a list, or an object of class "gpc.poly" (see package gpclib for a description of the class). Study areas described by multiple polygons must be described by lists or objects of class "gpc.poly". Objects of class "gpc.poly" can be constructed from matrices (single polygon) and lists (multiple polygons) with as(*object*,"gpc.poly").

```
> str(wica.gpc)
Formal class 'gpc.poly' [package "gpclib"] with 1 slots
  ..@ pts:List of 1
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:76] 632999 633008 633042 634179 634181 ...
  .. .. ..$ y   : num [1:76] 4832033 4831626 4831627 4831646 4831566 ...
  .. .. ..$ hole: logi FALSE
> str(hole.gpc)
Formal class 'gpc.poly' [package "gpclib"] with 1 slots
  ..@ pts:List of 1
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:10] 621309 622089 623798 624608 625207 ...
  .. .. ..$ y   : num [1:10] 4828695 4829655 4829715 4829085 4827316 ...
  .. .. ..$ hole: logi TRUE
> str(wica.hole.gpc)
```

```
Formal class 'gpc.poly' [package "gpclib"] with 1 slots
  ..@ pts:List of 2
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:76] 632999 633008 633042 634179 634181 ...
  .. .. ..$ y   : num [1:76] 4832033 4831626 4831627 4831646 4831566 ...
  .. .. ..$ hole: logi FALSE
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:10] 621309 622089 623798 624608 625207 ...
  .. .. ..$ y   : num [1:10] 4828695 4829655 4829715 4829085 4827316 ...
  .. .. ..$ hole: logi TRUE
```

Executing `hr.mcp` creates an object of class `"hr.mcp"`. Output includes `id`, an optional identifier for the estimate; `exclude`, the proportion of outlying locations to be excluded from calculations; an object of class `"gpc.poly"` that includes coordinates for vertices of an mcp home range; and `area`, area of the mcp (in squared axis units). By default (`plot=TRUE`), the function also produces a diagnostic plot (Fig. 1).

```
> mcp1 <- hr.mcp(x=xy.elk,id="Example 1")
> str(mcp1)
List of 5
 $ id     : chr "Example 1"
 $ exclude: num 0
 $ n      : Named num [1:4] 2300 0 0 2300
  ..- attr(*, "names")= chr [1:4] "total" "inadmissible" "excluded" "used"
 $ mcp    :Formal class 'gpc.poly' [package "gpclib"] with 1 slots
  .. ..@ pts:List of 1
  .. .. ..$ :List of 3
  .. .. .. ..$ x   : num [1:16] 625910 625743 622302 613286 611590 ...
  .. .. .. ..$ y   : num [1:16] 4821750 4821500 4817527 4816478 4821826 ...
  .. .. .. ..$ hole: logi FALSE
 $ area   : num 1.77e+08
 - attr(*, "class")= chr [1:2] "list" "hr.mcp"
```

Excluding influential or inadmissible values

Excluding observations from home range calculations is common practice and `hr.mcp` provides 3 options (Fig. 2):

1. Exclude a specified proportion of points that lie furthest from the spatial median of coordinates. This approach is the default because it is robust to the influence of far-flung points such as may arise from recording errors.

   ```
   > mcp2 <- hr.mcp(xy.elk,exclude=0.05)
   ```

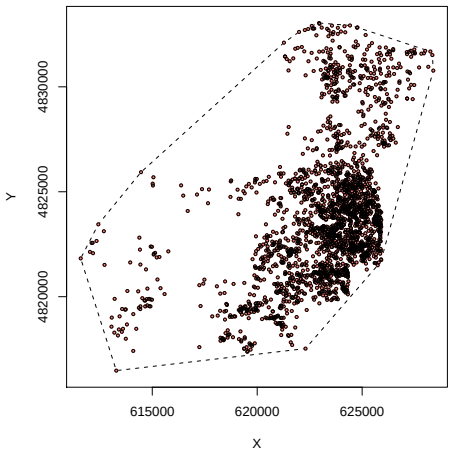

Figure 1: *Diagnostic plot produced by* `hr.mcp`.

2. Exclude points that lie furthest from the barycenter (arithmetic mean of coordinates). Implemented primarily for comparison with other packages.

```
> mcp3 <- hr.mcp(xy.elk,exclude=0.05,method="mean")
```

3. Exclude points that lie within inadmissible regions (e.g., a terrestrial mammal located in a lake or at sea).

```
> mcp4 <- hr.mcp(xy.elk,domain=wica.gpc)
```

Home range clipping

Once computed, home ranges can be "clipped" so that boundaries coincide with boundaries of polygons described by a matrix (single polygon), list, or object of class `"gpc.poly"`. In the following example, all of the data have been used to calculate a home range, but the result has been clipped so it does not extend beyond the boundary of Wind Cave National Park (Fig. 3).

```
> hr.mcp(xy.elk,clip.to=wica.gpc)
```

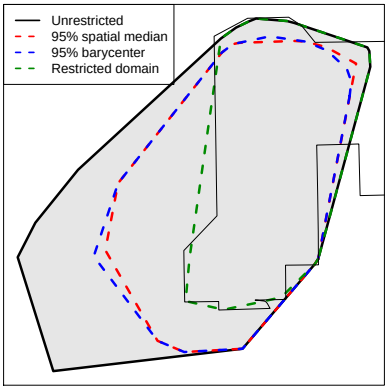

Figure 2: *Available criteria for identifying "outliers" and inadmissible points include distance from the spatial median, distance from the barycenter, and restriction of the domain to points within, e.g., Wind Cave National Park.*

Home range area

Components of home range objects can be extracted and used for subsequent calculations. For example, the component `'n'` provides information about sample sizes (number of data points, number removed by restricting the domain, number excluded by trimming outlying points, and number used in computations):

```
> mcp4$n
       total inadmissible     excluded         used
        2300          261            0         2039
```

The `area` component provides the home range area in squared axis units (converted from square meters to square kilometers by division in the example below).

```
> mcp4$area/(1000^2)
[1] 84
```

In many cases, study areas are not contiguous or enclose areas that are obviously inaccessible or unsuitable (e.g., lakes, towns, fenced areas). Investigators may wish to exclude such areas when computing estimates of home range size. If `clip.to` is specified, home range estimates are adjusted automatically.

Consider `wica.hole.gpc`, an object of class `"gpc.poly"` that describes 2 polygons, 1 of which represents a hole.

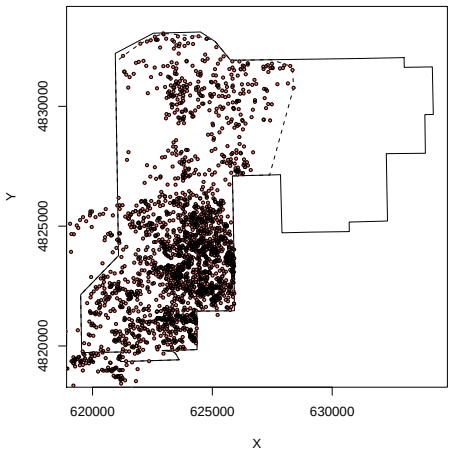

Figure 3: *An example of home range clipping.*

```
> data(wica.hole.gpc)
> str(wica.hole.gpc)
Formal class 'gpc.poly' [package "gpclib"] with 1 slots
  ..@ pts:List of 2
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:76] 632999 633008 633042 634179 634181 ...
  .. .. ..$ y   : num [1:76] 4832033 4831626 4831627 4831646 4831566 ...
  .. .. ..$ hole: logi FALSE
  .. ..$ :List of 3
  .. .. ..$ x   : num [1:10] 621309 622089 623798 624608 625207 ...
  .. .. ..$ y   : num [1:10] 4828695 4829655 4829715 4829085 4827316 ...
  .. .. ..$ hole: logi TRUE
```

Home range clipped to the park boundary:

```
> mcp5 <- hr.mcp(x=xy.elk,clip.to=wica.gpc)
```

Home range clipped to the park boundary, with the hole excluded:

```
> mcp6 <- hr.mcp(x=xy.elk,clip.to=wica.hole.gpc)
```

The following examples give area of a home range prior to clipping, area after clipping to the park boundary, and area after clipping and deduction of a hole:

```
> mcp1$area/(1000^2)
[1] 177
> mcp5$area/(1000^2)
[1] 76
```

```
> mcp6$area/(1000^2)
[1] 57
```

Repetitive operations

In practice, investigators often require estimates for numerous subsets of data. For example, estimates might be desired for many individuals, for a series of time periods, or both. One way to add such functionality–but at the expense of generality–is to provide additional arguments or more complex and restrictive input and output. The alternative (and the approach I've taken) is to promote generality and flexibility by relying, as much as possible, on existing functions like `for` and `lapply`. The following examples demonstrate.

Creates an ordered factor that groups the data into 2 subsets of approximately equal size:

```
> grouping.var <- cut(1:nrow(xy.elk),breaks=2,labels=1:2,
   ordered_result=TRUE)
> xy <- data.frame(group=grouping.var,xy.elk)
> head(xy)
  group      x       y
4     1 619347 4819457
5     1     NA      NA
6     1 618902 4819067
7     1 619727 4819281
8     1 619193 4819279
9     1 618553 4819621
```

Splits the data frame into a list with 1 element for each level of the grouping variable:

```
> groups <- split(xy,f=xy$group)
```

Computes a home range for each element of the list:

```
> hr.list1 <- lapply(groups,function(xy){
   with(xy,hr.mcp(x=x,y=y,clip.to=wica.gpc,plot=FALSE))
   })
```

Results are stored in a list; each component is an object of class `"hr.mcp"`:

```
> str(hr.list1,strict.width="cut")
List of 2
 $ 1:List of 5
  ..$ id     : logi NA
  ..$ exclude: num 0
```

```
  ..$ n      : Named num [1:4] 1131 0 0 1131
  .. ..- attr(*, "names")= chr [1:4] "total" "inadmissible..
  ..$ mcp    :Formal class 'gpc.poly' [package "gpclib"] w..
  .. .. ..@ pts:List of 1
  .. .. .. ..$ :List of 3
  .. .. .. .. ..$ x   : num [1:55] 624776 625074 625093 62..
  .. .. .. .. ..$ y   : num [1:55] 4832831 4832732 4832720..
  .. .. .. .. ..$ hole: logi FALSE
  ..$ area   : num 75501881
  ..- attr(*, "class")= chr [1:2] "list" "hr.mcp"
 $ 2:List of 5
  ..$ id     : logi NA
  ..$ exclude: num 0
  ..$ n      : Named num [1:4] 1169 0 0 1169
  .. ..- attr(*, "names")= chr [1:4] "total" "inadmissible..
  ..$ mcp    :Formal class 'gpc.poly' [package "gpclib"] w..
  .. .. ..@ pts:List of 1
  .. .. .. ..$ :List of 3
  .. .. .. .. ..$ x   : num [1:54] 625154 625763 627157 62..
  .. .. .. .. ..$ y   : num [1:54] 4832660 4831921 4831940..
  .. .. .. .. ..$ hole: logi FALSE
  ..$ area   : num 7.4e+07
  ..- attr(*, "class")= chr [1:2] "list" "hr.mcp"
```

A similar approach can be used to compute several types of estimates from the same data. For example, one can compute 75%, 95%, and 100% polygons in a single step:

```
> exclude.list <- list(0,0.05,0.25)
> names(exclude.list) <- c("Exclude 0", "Exclude 0.05", "Exclude 0.25")
> hr.list2 <- lapply(exclude.list,function(exclude){
   hr.mcp(xy.elk, exclude=exclude,plot=FALSE)
 })
```

Graphics

By design, `hr.mcp` does not support grooming of the plots it produces. Complete flexibility is provided instead by an associated plotting method (`plot.hr.mcp`), which is designed to integrate with the R graphics package. The function accepts optional arguments to `polygon` as well as the `xlim` and `ylim` arguments to `plot`. An example featuring numerous options and a supporting function (`point.in.hr`) for classifying points as "in" or "out" of a home range is implemented below and shown in Fig. 4.

Compute limits of a plotting region that will encompass the entire study area and all of the data:

```
> xlim <- range(c(xy.elk$x,get.bbox(wica.gpc)$x),na.rm=TRUE)
> ylim <- range(c(xy.elk$y,get.bbox(wica.gpc)$y),na.rm=TRUE)
```

Logical vectors identifying points as 1) outside the home range, 2) inside the home range, or 3) vertices of the home range:

```
> outside <- point.in.hr(x=xy.elk,hr=mcp2)==0
> inside <- point.in.hr(x=xy.elk,hr=mcp2)==1
> vertex <- point.in.hr(x=xy.elk,hr=mcp2)==3
```

Coordinated use of `plot.hr.mcp` and the R graphics package can produce virtually any desired result:

```
> plot(mcp2,lty=2,xlim=xlim,ylim=ylim,col="gray",id=FALSE)
> points(xy.elk[outside,],col="gray",cex=0.75,pch=19)
> points(xy.elk[inside,],col="tan1",cex=0.75,pch=19)
> points(xy.elk,cex=0.75)
> points(xy.elk[vertex,],col="red",cex=1.2,pch=17)
> points(xy.elk[vertex,],cex=1.2,pch=2)
> plot(wica.gpc,poly.args=list(lwd=2),add=TRUE)
> box()
```

Special considerations apply when home ranges include holes. Argument `'col'` will be disabled, shading of polygons will not be supported, and a warning will be issued. This restriction prevents overplotting from obscuring features of home ranges. Shading still can be accomplished with modest effort, by extracting component polygons and holes from home ranges and plotting them in a sensible order (Fig. 5):

```
> hole <- get.pts(mcp6$mcp)[[1]]
> poly <- get.pts(mcp6$mcp)[[2]]
> plot(mcp6)
> polygon(poly$x,poly$y,col="gray")
> polygon(hole$x,hole$y,col="white")
> box()
```

References

Roger D. Peng with contributions from Duncan Murdoch and Barry Rowlingson; GPC library by Alan Murta (2010). gpclib: General Polygon Clipping Library for R. R package version 1.5-1. http://CRAN.R-project.org/package=gpclib

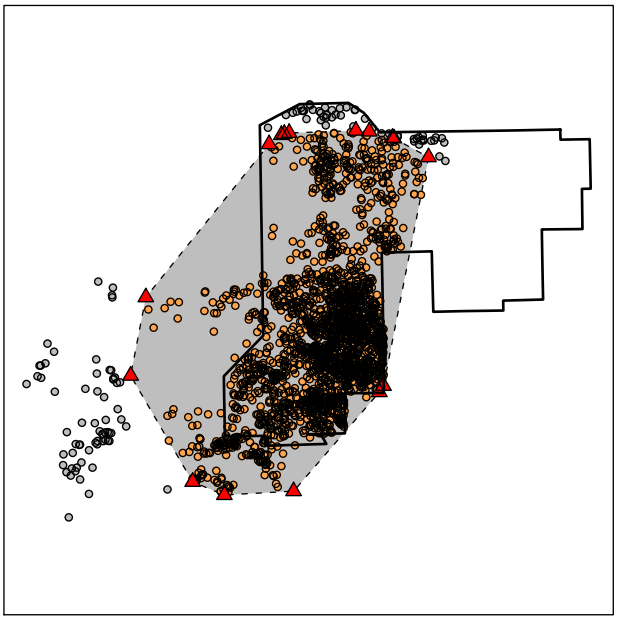

Figure 4: *A sample plot constructed with* `plot.hr.mcp` *and the R* graphics *package.*

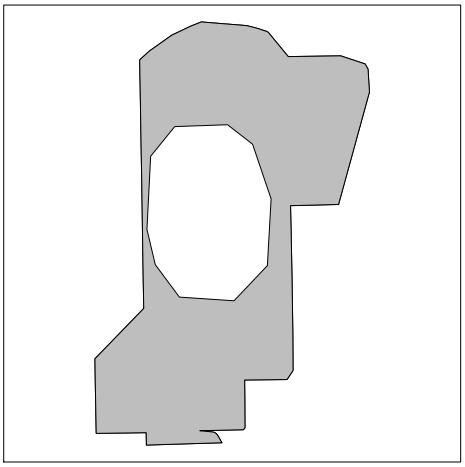

Figure 5: *When home ranges include holes, argument* `'col'` *is disabled; however, shading can still be accomplished by plotting home range components individually.*