Computing Hegyi's (and other) Competition Indices

siplab, Vignette #1

The competition index of Hegyi (1974) is one of the most popular in the forest modelling literature. I show here how to compute it with **siplab**. Pointers are also given for computing any other index based on pairwise tree interactions. These are the most common indices in individual-tree distance-dependent (*aka* spatially explicit) growth models, and can be handled by function `pairwise()`.

Data

If not already available, install **siplab** with `install.packages("siplab")` or through clicking on the appropriate GUI menus. This also installs the required **spatstat** package.

Let us use the `spruces` data set included in **spatstat**. It contains measurements for 134 trees in a 56×38 m sample plot. These data are already in the required `ppp` format, but for this example pretend that they are given as a simple data frame called `trees`:

```
library(siplab)

## Loading required package: spatstat
## Loading required package: spatstat.data
## Loading required package: spatstat.geom
## spatstat.geom 2.3-2
## Loading required package: spatstat.random
## spatstat.random 2.1-0
## Loading required package: spatstat.core
## Loading required package: nlme
## Loading required package: rpart
## spatstat.core 2.4-0
## Loading required package: spatstat.linnet
## spatstat.linnet 2.3-2
##
## spatstat 2.3-3       (nickname: 'That's not important right now')
## For an introduction to spatstat, type 'beginner'
```

```
##
## Attaching package: 'siplab'
## The following object is masked from 'package:spatstat.geom':
##
##     edges
```

```
trees <- as.data.frame(spruces)
summary(trees)
```

```
##        x               y              marks
##  Min.   : 0.70   Min.   : 1.200   Min.   :0.1600
##  1st Qu.:16.57   1st Qu.: 9.225   1st Qu.:0.2200
##  Median :30.40   Median :18.600   Median :0.2450
##  Mean   :29.57   Mean   :18.734   Mean   :0.2504
##  3rd Qu.:43.88   3rd Qu.:27.975   3rd Qu.:0.2700
##  Max.   :55.00   Max.   :36.600   Max.   :0.3700
```

Typically, the data would have been read from a data file by a command such as `trees <- read.csv("data.csv")`. The `x` and `y` columns are tree coordinates, in meters, and `marks` are diameters at breast height (dbh), also in meters. For a more conventional appearance, rename and convert the diameters to centimeters:

```
names(trees)[3] <- "dbh"
trees$dbh <- trees$dbh * 100
head(trees)
```

```
##     x    y dbh
## 1 2.4  1.4  21
## 2 1.9  3.3  25
## 3 0.7 14.4  28
## 4 2.1 17.2  23
## 5 2.5 25.1  25
## 6 3.4 26.5  22
```

Now, for computations in **siplab**, we need to convert (back) the data to the form of a **spatstat** point pattern object of class `ppp`. This is done with function `ppp()`, passing as arguments the coordinate vectors, the plot dimensions ("window"), and the marks (dbh):

```
trees <- ppp(trees$x, trees$y, c(0,56), c(0,38), marks=trees$dbh)
summary(trees)
```

```
## Marked planar point pattern:  134 points
## Average intensity 0.06296992 points per square unit
##
## Coordinates are given to 1 decimal place
```

```
## i.e. rounded to the nearest multiple of 0.1 units
##
## marks are numeric, of type 'double'
## Summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   22.00   24.50   25.04   27.00   37.00
##
## Window: rectangle = [0, 56] x [0, 38] units
## Window area = 2128 square units
```

The plot has been specified by the x-range 0 to 56 and the y-range 0 to 38. It is also possible to have more than one mark, for instance, diameter and height in the data set `finpines`. In that case `marks` is a data frame. In addition to dimensions, it may be useful to include, for instance, tree and species codes.

An alternative is to use instead `trees <- as.ppp(trees, c(0,56.0,38))`.

The index

The commonly used form of Hegyi's index for a target tree i is

$$C_i = \sum_j \frac{D_j/D_i}{R_{ij}} \,, \tag{1}$$

where D_i and D_j are the dbh of the target tree and the dbh of competitor j, respectively, and R_{ij} is the distance between trees i and j. The sum is over the "competitors" j of tree i. In the case of Hegyi's index, competitors are defined as all the trees that are within a given distance R_{max} from the target tree.

This is an example of the class of competition indices handled by function `pairwise()`. These are based on some function of sizes and distance for pairs of trees (hence "pairwise"). In **siplab** the function is called a *kernel*. The kernel values for the pairs formed by the target tree and each of its competitors are added up. In general, competitors may be defined as those trees within some radius around the target, as in this instance (`maxR` = R_{max}), or as the n nearest neighbors (`maxN` = n), or in some other way through a `select` function.

Here we can use a built-in kernel called `powers_ker()`, which represents the general form

$$\frac{S_j^{p_j}/S_i^{p_i}}{R_{ij}^{p_r}} \,, \tag{2}$$

where S is some measure of size. Hegyi's index corresponds to the special case $p_i = p_j = p_r = 1$. Type `?kernel` for other built-ins or for how to write your own.

Thus, choosing 6 meters for R_{max}, the index can be obtained from

```
hegyi <- pairwise(trees, maxR=6, kernel=powers_ker,
    kerpar=list(pi=1, pj=1, pr=1, smark=1))
```

The list `kerpar` specifies the kernel parameter values and the size variable. That is, p_i, p_j, p_r in (2), and the first (and only) mark. Actually, `kerpar` could be omitted here because these values happen to be the default for `powers_ker()`. The computed indices are returned as an additional column `cindex` in the marks:

```
head(marks(hegyi))
```

```
##   mark    cindex
## 1   21 0.6059364
## 2   25 0.4275487
## 3   28 0.5569033
## 4   23 1.0812680
## 5   25 1.4489832
## 6   22 2.0487122
```

In the original article, Hegyi added 1 foot to the distance in the denominator (Hegyi, 1974). See the Example in `?kernel` for a kernel function specific to this case. Use it as `pairwise(trees, maxR=6, kernel=hegyiorig_ker)`, no `kerpar` needed.

Edge effects

Trees near edges tend to have abnormally low competition indices because competitors outside the plot are missing. Therefore, one may want to exclude such trees from the results. The `edges()` function can be used to keep only trees not closer than a given distance from the edges, e.g., 6 m:

```
hegyi_trim <- edges(hegyi, -6)
summary(hegyi_trim)
```

```
## Marked planar point pattern:  77 points
## Average intensity 0.06730769 points per square unit
##
## Coordinates are given to 1 decimal place
## i.e. rounded to the nearest multiple of 0.1 units
##
## Mark variables: mark, cindex
## Summary:
##       mark           cindex
##  Min.   :16.00   Min.   :0.417
##  1st Qu.:22.00   1st Qu.:1.355
##  Median :24.00   Median :1.763
```

```
##  Mean   :24.74   Mean   :1.756
##  3rd Qu.:26.00   3rd Qu.:2.200
##  Max.   :37.00   Max.   :3.511
##
## Window: rectangle = [6, 50] x [6, 32] units
##                     (44 x 26 units)
## Window area = 1144 square units
```

An alternative edge correction method starts by faking a surround through translations of the real plot. Only part of the expansion is usually needed, in this instance up to 6 m around the real plot. This can be done with `edges(trees, 6)`. After computing the competition indices in the expanded plot, the result is trimmed to the original plot size with `edges(hegyi, -6)`. See `?edges` for details.

References

Hegyi, Frank. (1974) *A Simulation Model for Managing Jack-pine Stands*. P. 74–90 in Fries, J. (Ed.) *Growth Models for Tree and Stand Simulation*. Royal College of Forestry, Department of Forest Yield Research, Research Notes 30. Stockholm, Sweden.

March 7, 2022