Package 'siplab'

September 4, 2020

Title Spatial Individual-Plant Modelling

Version 1.5

Author Oscar Garcia <https://orcid.org/0000-0002-8995-1341>

Maintainer Oscar Garcia <garcia@dasometrics.net>

Description A platform for computing competition indices and experimenting with spatially explicit individual-based vegetation models.

Depends spatstat (>= 1.36-0)

License GPL (>= 2)

LazyData true

Suggests knitr

VignetteBuilder knitr

URL https://github.com/ogarciav/siplab/

BugReports https://github.com/ogarciav/siplab/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2020-09-04 12:20:06 UTC

R topics documented:

siplab-package *Spatial Individual-Plant Simulation*

Description

A platform for experimenting with spatially explicit individual-based plant modelling

Details

Package:	siplab
Type:	Package
Version:	1.5
License:	GPL

The main top level functions are `pairwise()`, and `assimilation()`.

`pairwise()` computes the competition indices most commonly used in individual-tree distance-dependent (or spatially explicit) forest growth models. These indices are based on a sum of functions of size and distance between the subject plant and each of its competitors. They represent an aggregate of pairwise interactions, the angular configuration of competitors and any higher-order interactions are ignored. Each index is characterized by a specific interaction function, here called a kernel, and by a definition of competitors.

`assimilation()` deals with "fully spatial" models, computing "assimilation indices" that aim at a mechanistic approximation of effective resource capture. One starts with a spatial resource distribution that is typically assumed to be uniform, Plants exert competitive pressure depending on size and distance, described by influence functions. The resource available at each point is allocated to plants according to their local influence and to a partition rule. Finally, the resource uptake may be weighted by an efficiency function that depends on size and distance, and is spatially integrated to obtain the plant's assimilation index. Several examples of influence and efficiency functions are pre-programmed, and others can be easily produced.

The `edges()` function is useful for handling edge effects.

Some sample data sets are included, see links below.

The package is built on top of the **spatstat** library (`http://spatstat.org/`), which needs to be installed first.

Author(s)

Oscar García

Maintainer: O. Garcia <garcia@dasometrics.net>

References

García, O. "Siplab, a spatial individual-based plant modelling system". Computational Ecology and Software 4(4), 215-222. 2014. (`http://www.iaees.org/publications/journals/ces/articles/2014-4(4)/2014-4(4).asp`).

García, O. "A generic approach to spatial individual-based modelling and simulation of plant communities". Mathematical and Computational Forestry and Nat.-Res. Sci. (MCFNS) 6(1), 36-47. 2014. (http://mcfns.net/index.php/Journal/article/view/6_36).

https://github.com/ogarciav/siplab.

http://forestgrowth.unbc.ca/siplab/ (no longer maintained).

See Also

Example **siplab** data sets: boreasNP, boreasNS, boreasSA, boreasSP.

Some **spatstat** standard data sets may also be of interest: finpines, longleaf, spruces, waka.

For tutorials try the vignettes. E. g., in R type help.start() to open the help browser, and navigate to Packages > siplab > Vignettes.

Examples

```
# Pretend that the data is given as a simple data frame
data <- as.data.frame(spruces) #  from a spatstat data set
head(data) # x-y coordinates in a 56x38 m plot, marks are dbh in meters
data$marks = data$marks * 100  # dbh in cm
# Convert to a point pattern object
datap <- as.ppp(data, c(0, 56, 0, 38))  # plot limits (minx, maxx, miny, maxy)
# or datap <- ppp(data$x, data$y, c(0, 56), c(0, 38), marks = data$marks)
# Hegyi (1974) index (as usual without his original 1-foot distance offset)
hegyi <- pairwise(datap, maxR = 6, kernel = powers_ker, kerpar = list(pi=1,
    pj=1, pr=1, smark=1))
head(marks(hegyi))
# ZOI model
zoi <- assimilation(datap, influence=zoi_inf, infpar=c(k=0.2, smark=1),
    asym=1)
```

---

assimilation *Compute Assimilation Indices*

Description

This is the main function in **siplab** for computing assimilation indices. Optionally, it computes also a free-growing index, and/or the assimilation centroid.

Usage

```
assimilation(plants, pixsize = 0.2, resource = 1, influence =
    gnomon_inf, infpar = NULL, asym = Inf, efficiency = flat_eff,
    effpar = NULL, plot = TRUE, afree = FALSE, centroid = FALSE)

assimilation_pix(plants, pixsize = 0.2, resource = 1, influence =
    gnomon_inf, infpar = NULL, asym = Inf, efficiency = flat_eff,
    effpar = NULL, plot = TRUE, afree = FALSE, centroid = FALSE)
```

Arguments

plants	A **spatstat** point pattern object (class ppp). It contains the plants coordinates, and marks with the plant size and possibly other attributes.
pixsize	Resolution, approximate step size in the pixel grid. Default 0.2.
resource	Either a pixel image (class im), or a function, or other object that can be converted to a pixel image, specifying the spatial distribution of resource availability. If an image, it should cover the plants window. It is adjusted to the plants window size and specified resolution if necessary. Default is 1, a uniform distribution with 1 unit of resource per unit area.
influence	Function for computing influence values. Must take arguments (dx,dy,marks,par), where dx is a vector of points-to plant x-distances, dy is a vector of points-to plant y-distances, marks are the plant marks, and par receives the value of the infpar argument. Examples are provided in the functions tass_inf(), etc. (see influence). Default: gnomon_inf.
infpar	Parameter(s) for influence, a list or vector. Default: list(a=1,b=4,smark=1). Here smark=1 indicates that the plant size variable is the first or only item in marks.
asym	Asymmetry parameter α in the allotment function. Default is Inf, which corresponds to one-sided competition (tesselation models). In old versions of **siplab** this parameter was called partpar.
efficiency	Efficiency function for weighting the point-wise resource uptake. Must take arguments (dx,dy,marks) or (dx,dy,marks,par), where dx is a vector of points-to plant x-distances, dy is a vector of points-to plant y-distances, marks are the plant marks, and par receives the value of the effpar argument if not NULL. Examples are provided in the functions tass_eff(), etc. (see efficiency). The default is flat_eff, no weighting.
effpar	Parameter(s) for efficiency, usually a list or vector. Default: NULL.
plot	If TRUE, the denominator of the allotment function is graphed as a pixel image, to visualize competition pressure (default).
afree	If TRUE, the free-growing assimilation is also computed. Default is FALSE.
centroid	If TRUE, the centroid of the plant assimilation distribution is also computed. Default is FALSE.

Details

assimilation() and assimilation_pix() are functionally equivalent, but the code in assimilation_pix() is somewhat clearer and slower. It may be useful for documentation purposes, and as a basis for user modification.

Computation starts with a resource intensity map at a spatial resolution given by pixsize. Typically the resource distribution is assumed to be uniform (the default). Plants exert competitive pressure depending on size and distance, described by the influence function. The resource available at each pixel is allotted to plants according to their influence and to an allotment rule parametrized by asym. Finally, the resource uptake is weighted by the efficiency function, and is spatially integrated to obtain the plant's assimilation index. Besides size, influence and efficiency functions can include other plant attributes, such as species.

Value

Returns the input point pattern `plants`, with the marks replaced by a data frame containing the original marks followed by one or more columns containing the computed results. The additional column are the assimilation indices in column `aindex`, and optionally the free-growing index in `afree`, and/or the x and y centroid coordinates in `cx` and `cy`.

Note

Requires the **spatstat** package.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab.

García, O. (2013) "A generic approach to spatial individual-based modelling and simulation of plant communities". Mathematical and Computational Forestry and Nat.-Res. Sci. (MCFNS) 6(1), 36-47. 2014.

See Also

`influence`, `efficiency`, `edges`

Examples

```
a <- assimilation(finpines, infpar=list(a=1, b=4,
     smark="height"), afree=TRUE)
summary(a)
system.time(assimilation_pix(finpines))
system.time(assimilation(finpines))
```

`boreas` *Marked Point Pattern Tree Data from BOREAS*

Description

Four data sets from the Boreal Ecosystem–Atmosphere Study (BOREAS, Rich and Fournier 1999), as used by García (2006). These are approximately evenaged and single-species unmanaged natural forests, from a northern study area in Manitoba and a southern study area in Saskatchewan, central Canada. Tree coordinates and diameters at breast height (dbh) were measured for all trees taller than 2 m on areas of 50 m $\times$ 60 m, subdivided into subplots on a 10 m grid. Tree heights were estimated from height-dbh regressions based on a sample of height measurements. The data here excludes dead trees, and also excludes some trees with coordinates just outside the observation window.

The 4 data sets are:

`boreasNP`**:** Northern study area, Jack pine

`boreasNS`: Northern study area, black spruce

`boreasSA`: Southern study area, trembling aspen

`boreasSP`: Southern study area, Jack pine

Format

Each data set is a **spatstat** marked point pattern object (class ppp). The marks are a data frame with dbh (cm), `height` (m), `species`, a `dominance` classification, and a subplot id.

Source

https://doi.org/10.3334/ORNLDAAC/359

References

Rich, P.M., and Fournier, R. (1999) BOREAS TE-23 map plot data [online]. Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee. Available from http://daac.ornl.gov.

García, O. (2006) Scale and spatial structure effects on tree size distributions: Implications for growth and yield modelling. *Canadian Journal of Forest Research* **36**(11), 2983–2993. https://www.researchgate.net/publication/237866519.

Examples

```
summary(boreasNP)
plot(boreasNP)
## Not run: aNP <- assimilation(boreasNP)
# this may take a few minutes!
```

---

edges *Adjust for Edge Effects*

Description

Shrink a point pattern, or expand it through replication.

Usage

```
edges(plants, width)

core(plants, distance)
```

Arguments

`plants`	A **spatstat** point pattern object (class ppp). It normally contains the plants coordinates, and marks with the plant size and possibly other attributes.
`width`	Distance from the edges to shrink, if negative, or to expand, if positive.
`distance`	Distance from the edges.

Details

When computing assimilation or competition indices, those near the edges of the study region are distorted because the outside is empty. Common solutions to this problem are not to use indices computed for plants near the edges, or (with rectangular regions) to attach translated copies, thus changing the topology intothat of a torus. This function implements both strategies. When expanding, the extentt of the copies to be used can be specified to avoid unnecessary computation.

Typically, in the first case the indices are computed for the full pattern, and then the edges are discarded using `edges()` with a negative `width`. In the second case, the point pattern is first expanded with `edges(plants,width)`, the indices are computed for the expanded pattern, and then the resulta are restricted to the original size with `edges(result,-width)`.

`core()` returns a logical vector indicating which plants are at more than the given distance from the edges. Thus, `plants[core(plants,width)]` is equivalent to `edges(plants,-width)`.

Value

`edges()` returns a point pattern with the same structure as `plants`.

If `width` is negative, the parts of the pattern that are at a distance less than `-width` from an edge are discarded.

If `width` is positive, the pattern is first expanded by surrounding it with 8 shifted copies (the window must be rectangular). Then, the parts of the pattern that are at a distance greater than `width` from an edge of the original pattern are discarded.

If `width` = 0, `plants` is returned unchanged.

`core()` returns a logical vector with `TRUE` for the plants that are at more than the given distance from the edges, and `FALSE` for the rest.

Note

Requires the **spatstat** package.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

See Also

assimilation, pairwise

Examples

```
finpines
edges(finpines, 3)
edges(finpines, -3)
```

efficiency *Efficiency Functions*

Description

Compute efficiency values depending on distance and plant marks, for use in assimilation().

Note: In previous versions of **siplab** the function names had .eff in place of _eff.

Usage

```
flat_eff(dx, dy, marks)

tass_eff(dx, dy, marks, par = list(b = 3.52 * 0.975, c = 6.1,
    smark = 1))

gates_eff(dx, dy, marks, par = list(a = 1, b = 4, smark = 1))

gnomon_eff(dx, dy, marks, par = list(a = 1, b = 4, smark = 1))
```

Arguments

dx	Vector of x-distances. Coordinates x for a number of points minus coordinate x of the subject plant.
dy	Vector of y-distances. Coordinates y for a number of points minus coordinate y of the subject plant.
marks	Mark data for the subject plant.
par	Optional vector or list of parameters.

Details

The user can program her/his own efficiency function. It must take the arguments dx, dy, marks, and optionally par.

Efficiency function values are normally non-negative. Otherwise, they are set to 0 in assimilation().

The values of par are taken from the argument effpar of assimilation(), if not NULL. Otherwise the default is used.

smark in par must be 1 or "mark" if there is only one mark. If the marks are a data frame, smark must be the number or name of the column with the plant size variable.

flat_eff() returns 1, independently of plant size or distance.

tass_eff(), gates_eff(), and gnomon_eff() are proportional to their influence function counterparts (see influence), scaled to be 1 at the origin.

Value

Vector of efficiency values, of length equal to the length of dx and dy.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

García, O. "A generic approach to spatial individual-based modelling and simulation of plant communities". Mathematical and Computational Forestry and Nat.-Res. Sci. (MCFNS) 6(1), 36-47. 2014.

See Also

assimilation, influence

Examples

```
# Example multi-species efficiency function (spruce/hardwoods)
multi_eff <- function (dx, dy, marks, par) {
    out <- numeric(length(dx))
    s <- marks$SPECIES == "Spruce"
    out[s] <- gnomon_eff(dx[s], dy[s], marks[s, ], par=list(a=par$aS,
        b=par$bS, smark=par$smark))
    out[!s] <- gnomon_eff(dx[!s], dy[!s], marks[!s, ], par=list(a=par$aH,
        b=par$bH, smark=par$smark)) # Hardwoods
    return(out)
}
```

---

influence *Influence Functions*

Description

Compute influence values depending on distance and plant marks, for use in assimilation().

Note: In previous versions of **siplab** the function names had .inf in place of _inf.

Usage

```
zoi_inf(dx, dy, marks, par = list(k = 0.2, smark = 1))

tass_inf(dx, dy, marks, par = list(b = 3.52 * 0.975, c = 6.1,
    smark = 1))

gates_inf(dx, dy, marks, par = list(a = 1, b = 4, smark = 1))

gnomon_inf(dx, dy, marks, par = list(a = 1, b = 4, smark = 1))
```

Arguments

dx	Vector of x-distances. Coordinates x for a number of points minus coordinate x of the subjectt plant.
dy	Vector of y-distances. Coordinates y for a number of points minus coordinate y of the subjectt plant.
marks	Mark data for the subject plant.
par	List of parameters.

Details

The user can program her/his own influence function. It must take the arguments dx, dy, marks, and optionally par.

Influence function values are normally non-negative. Otherwise, they are set to 0 in assimilation().

The values of par are taken from the argument infpar of assimilation(), if not NULL. Otherwise the default is used.

smark in par must be 1 or "mark" if there is only one mark. If the marks are a data frame, smark must be the number or name of the column with the plant size variable.

Let S be the plant size, and R be the Euclidean plant-to-point distance $R = \sqrt{\mathrm{dx}^2 + \mathrm{dy}^2}$. Then the built-in influence functions are:

zoi_inf(): 1 if $R < kS$, 0 otherwise

tass_inf(): $\max\{0,\ S - c[\exp(R/b) - 1]\}$

gates_inf(): $\max\{0,\ [S^a - (bR)^a]^{1/a}\}$

gnomon_inf(): $\max\{0,\ S - bR^a\}$

Other influence functions can be written following these examples.

Value

Vector of influence values, of length equal to the length of dx and dy.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

García, O. "A generic approach to spatial individual-based modelling and simulation of plant communities". Mathematical and Computational Forestry and Nat.-Res. Sci. (MCFNS) 6(1), 36-47. 2014.

See Also

assimilation

Examples

```
# Example multi-species influence function (spruce/hardwoods)
multi_inf <- function (dx, dy, marks, par) {
    out <- numeric(length(dx))
    s <- marks$SPECIES == "Spruce"
    out[s] <- gnomon_inf(dx[s], dy[s], marks[s, ], par=list(a=par$aS,
        b=par$bS, smark=par$smark))
    out[!s] <- gnomon_inf(dx[!s], dy[!s], marks[!s, ], par=list(a=par$aH,
        b=par$bH, smark=par$smark)) # Hardwoods
    return(out)
}
```

`kernel` *Competition Kernel Functions*

Description

Functions representing the effect of a competitor on a subject plant, depending on distance and plant sizes (marks). For use in `pairwise()`.

Note: In previous versions of **siplab** the function names had `.ker` in place of `_ker`.

Usage

```
powers_ker(imarks, jmarks, dists, dranks, par = list(pi=1, pj=1,
    pr=1, smark = 1))

staebler_ker(imarks, jmarks, dists, dranks, par = list(k=0.1, p=1,
    smark=1))

spurr_ker(imarks, jmarks, dists, dranks, par = list(type=1,
    smark=1))
```

Arguments

`imarks`	Marks for the subject plant, a 1-row data frame.
`jmarks`	Data frame with marks for competitors
`dists`	Vector of distances between the subject plant and the competitors.
`dranks`	Distance ranks.
`par`	List of parameters.

Details

The values of `par` are taken from the argument `kerpar` of `pairwise()`, if not `NULL`.

`smark` in `par` must be 1 or "mark" if there is only one mark. If the marks are a data frame, `smark` must be the number or name of the column with the plant size variable.

`powers_ker()` is a general form that includes many examples from the literature. If S_i is the size of the subject plant, S_j the size of the competitor, and R is the distance between them, then this kernel is $(S_j^{p_j}/S_i^{p_i})/R^{p_r}$. For instance, the popular Hegyi's index corresponds to `pi=1,pj=1,pr=1`. This and other examples could be coded directly if computational efficiency is important, see the example below.

`staebler_ker()` is the width of the overlap of zones of influence (ZOI), used by Staebler in 1951. Assumes that the ZOI radius is kS^p, where S is size.

`spurr_ker()` is an example of an index that depends on distance ranks: equations (9.5a), (9.5b) of Burkhart and Tomé (2012).

Competition kernels seem to be limited only by the researchers imagination. Others can be written following these examples.

Value

Vector of length equal to the length of `dists`.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

Burkhart, H. E. and Tomé, M. (2012) *Modeling Forest Trees and Stands*. Springer.

García, O. "Siplab, a spatial individual-based plant modelling system". Computational Ecology and Software 4(4), 215-222. 2014.

See Also

`pairwise`

Examples

```
# Originally Hegyi added one foot to the distance:
hegyiorig_ker <- function(imarks, jmarks, dists, ...) {
# Assume coordinates in meters, and a single mark (dbh)
    (jmarks$mark / imarks$mark) / (dists + 0.30481)
}
```

pairwise *Compute Pairwise Competition Indices*

Description

This function computes competition indices based on pairs of plants, ignoring higher-order interactions.

Usage

```
pairwise(plants, maxN = NULL, maxR = NULL, select = NULL, selpar =
         NULL, kernel, kerpar = NULL)
```

Arguments

plants	A **spatstat** point pattern object (class ppp). It contains the plants coordinates, and marks with the plant size and possibly other attributes.
maxN	Maximum number of nearest neighbors to include as potential competitors. Default is NULL (no restriction).
maxR	Maximum radius to search for potential competitors. Default is NULL (no restriction).
select	Optional user-supplied selection function for choosing competitors. Must take arguments (imarks,jmarks,dists,dranks) or (imarks,jmarks,dists,dranks,par), where imarks are the marks for the subject plant (a 1-row data frame), jmarks is a data frame with the marks of the potential competitors, dists is a vector distances between subject plant and the potential competitors, dranks are the distance ranks, and par receives the value of the selpar argument if not NULL. It must return a logical vector with the same length as dists. Examples are provided in the functions powlinear_sel(), etc. (see select). Default is NULL (no selection).
selpar	Parameter(s) for select, usually a list or vector. Default: NULL.
kernel	Competition kernel function for computing the effect of competitor j on the subject plant i. Must yake arguments (imarks,jmarks,dists,dranks) or (imarks,jmarks,dists,dranks,par), where imarks are the marks for the subject plant (a 1-row data frame), jmarks is a data frame with the marks of the potential competitors, dists is a vector of distances between subject plant and the potential competitors, dranks are the distance ranks, and par receives the value of the kerpar argument if not NULL. It must return a numeric vector with the same length as dists. Examples are provided in the functions powers_ker(), etc. (see kernel).
kerpar	Parameter(s) for kernel, usually a list or vector. Default: NULL.

Details

Traditionally, the competition index for a subject plant i is obtained in two stages: (1) Choose a set of competitors of i by some selection rule. (2) Compute a measure of the effect of each competitor j on plant i, and add over j. This effect of j on i is normally a function of the sizes of both plants and of the distance between them, which we call a competition kernel. The kernel may depend on other plant attributes, like species, and in some rare instances on the distance ranks or on the number of competitors. Conceptually, the first stage is not strictly necessary, it could be replaced by specifying zero kernel values (the effect of the selection is usually to truncate the kernel function beyond some distance). However, a separate selection rule may be more transparent, and may reduce the computational effort in searching for neighbors.

Some simple selection rules can be implemented by giving a value to `maxN` and/or `maxR`. In any case, reasonable limits on these variables may be advisable for reducing computation.

More complex rules can be specified by the `select` function, with parameters in `selpar`. See `select` for examples. If more than one of `maxN`, `maxR` or `select` are given, the intersection of the selection criteria is used.

Kernel computation is specified by the `kernel` function and the parameters in `kerpar`. See `kernel` for examples.

Value

Returns the input point pattern `plants`, with the marks replaced by a data frame containing the original marks followed by the competition index in a column named `cindex`.

Note

Requires the **spatstat** package.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

García, O. "Siplab, a spatial individual-based plant modelling system". Computational Ecology and Software 4(4), 215-222. 2014.

See Also

`select`, `kernel`, `edges`

Examples

```
# Hegyi (1974) index (no distance offset, as usual)
summary(pairwise(finpines, maxR = 6, kernel=powers_ker))
```

select *Competitor Selection Rules*

Description

Functions returning TRUE for plants that compete with a given subject plant, or FALSE otherwise. The decision can depend on distance and plant marks. For use in `pairwise()`.

Note: In previous versions of **siplab** the function names had `.sel` in place of `_sel`.

Usage

```
size_sel(imarks, jmarks, dists, dranks, par = list(k = 0.2, smark
    = 1))

powlinear_sel(imarks, jmarks, dists, dranks, par = list(ki = 0.2,
    kj = 0, p = 1, r0 = 0, smark=1))
```

Arguments

`imarks`	Marks for the subject plant, a 1-row data frame.
`jmarks`	Data frame with marks for potential competitors
`dists`	Vector of distances between the subject plant and the potential competitors.
`dranks`	Distance ranks.
`par`	List of parameters.

Details

The values of `par` are taken from the argument `selpar` of `pairwise()`, if not `NULL`.

`smark` in `par` must be 1 or "mark" if there is only one mark. If the marks are a data frame, `smark` must be the number or name of the column with the plant size variable.

`size_sel()` is a simple example where competitors are selected within a radius proportional to plant size. This corresponds to the second example in Section 9.2.1 of Burkhart and Tomé (2012).

Note that their first example (fixed radius) is implemented by giving a value to `maxR` in `pairwise()`, no `select` function is needed. Similarly, their third example (fixed number of nearest neighbors) is obtained by giving a value to `maxN`.

`powlinear_sel()` is a general form that covers all the other examples in Burkhart and Tomé (2012) by choosing specific parameters values (except for the *competition elimination angle*, which depends on relative positions among competitors and not only on distances). It implements a condition `distance < ki * sizei^p + kj * sizej^p + r0`, with the following special cases:

Multiple of crown radius: `kj=0,p=1,r0=0,smark="crownwidth"`.

Angle count sampling: `ki=0,p=1,r0=0,smark="dbh"`.

Areas of influence overlap: `ki=kj,p=1,r0=0`, if the radius is a linear function of size (`p` not 1 for an allometric relationship).

Vertical search cone: If the height of the cone vertex is constant, proportional to tree height, or more generally some linear function $c_1 h_i + c_2$, then `ki` = $-c_1/\tan(90-\beta/2)$, `kj` = $1/\tan(90-\beta/2)$, `p=1`, `r0` = $-c_2/\tan(90-\beta/2)$, `smark="height"`.

These and other examples could be coded directly if computational efficiency is important.

Value

Logical vector of length equal to the length of `dists`.

Author(s)

Oscar García.

References

https://github.com/ogarciav/siplab

Burkhart, H. E. and Tomé, M. (2012) *Modeling Forest Trees and Stands*. Springer.

See Also

pairwise

Index