

# Package ‘siplab’

January 10, 2014

**Type** Package

**Title** Spatial individual-plant modelling

**Version** 1.0

**Date** 2014-01-10

**Author** Oscar Garcia

**Maintainer** Oscar Garcia <garcia@unbc.ca>

**Description** A platform for experimenting with spatially explicit individual-based vegetation models.

**License** GPL (>= 2)

**Depends** spatstat

**URL** <http://forestgrowth.unbc.ca/siplab>

## R topics documented:

|                          |    |
|--------------------------|----|
| siplab-package . . . . . | 2  |
| assimilation . . . . .   | 3  |
| boreas . . . . .         | 5  |
| edges . . . . .          | 6  |
| efficiency . . . . .     | 8  |
| influence . . . . .      | 9  |
| kernel . . . . .         | 10 |
| pairwise . . . . .       | 11 |
| select . . . . .         | 13 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>15</b> |
|--------------|-----------|

---

 siplab-package

*Spatial Individual-Plant Simulation*


---

## Description

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

## Details

Package: siplab  
 Type: Package  
 Version: 1.0  
 Date: 2014-01-09  
 License: GPL

The main function is [assimilation](#), which computes effective resource capture (or “assimilation”) indices. 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 is 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, see [influence](#) and [efficiency](#). Some sample data sets are included, see links below.

Another class of spatial individual-plant models consider only pair-wise interactions between a plant and each of its neighbors, ignoring higher-order interactions. These are implemented in function [pairwise](#).

The [edges](#) function is useful for handling edge effects.

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

## Warning

Limited testing so far, use with care. Please email issues to <[garcia@unbc.ca](mailto:garcia@unbc.ca)>.

## Author(s)

Oscar García

Maintainer: O. Garcia <[garcia@unbc.ca](mailto:garcia@unbc.ca)>

## References

<http://forestgrowth.unbc.ca/siplab>

García, O. “A generic approach to spatial individual-based modelling and simulation of plant communities”. Manuscript in review.

**See Also**

Example **siplab** data sets: [boreasNP](#), [boreasNS](#), [boreasSA](#), [boreasSP](#).

Some **spatstat** standard data sets may also be of interest: [finpines](#), [longleaf](#), [spruces](#), [waka](#).

**Examples**

```
a <- assimilation(finpines, influence=zoi.inf, infpar=c(k=0.2,
  smark=1), partpar=1)
head(marks(a))
```

---

 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 = 0.1, resource = 1, influence = gnomon.inf,
  infpar = list(a = 1, b = 4, smark = 1), partpar = Inf, efficiency = flat.eff,
  effpar = NULL, plot = TRUE, afree = FALSE, centroid = FALSE)
```

```
assimilation.pix(plants, pixsize = 0.1, resource = 1, influence = gnomon.inf,
  infpar = list(a = 1, b = 4, smark = 1), partpar = Inf, efficiency = flat.eff,
  effpar = NULL, plot = TRUE, afree = FALSE, centroid = FALSE)
```

**Arguments**

|           |   |
|-----------|---|
| plants    | A <b>spatstat</b> point pattern object (class <code>ppp</code> ), containing the plants coordinates and marks with the plant size and possibly other attributes.  |
| pixsize   | Resolution, approximate step size in the pixel grid. Default 0.1.   |
| resource  | Either a pixel image (class <code>im</code> ), 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 | User-supplied influence function for computing influence values. Must have arguments <code>(dx, dy, marks, par)</code> , where <code>dx</code> is a vector of points-to plant x-distances, <code>dy</code> is a vector of points-to plant y-distances, <code>marks</code> are the plant marks, and <code>par</code> receives the value of the <code>infpar</code> argument. Examples are provided in the functions <code>*.inf</code> ( <a href="#">tass.inf</a> , etc.). Default: <a href="#">gnomon.inf</a> . |

|                         |  |
|-------------------------|--|
| <code>infpar</code>     | Parameter(s) for influence, usually a list or vector. Default: <code>list(a=1, b=4, smark=1)</code> . Note that <code>smark=1</code> indicates that the plant size variable is the first or only item in marks.  |
| <code>partpar</code>    | Exponent $\theta$ in the partition function. Default is <code>Inf</code> , which produces a tessellation.  |
| <code>efficiency</code> | User-supplied efficiency function for weighting the point resource uptake. Must have arguments ( <code>dx</code> , <code>dy</code> , <code>marks</code> , <code>par</code> ), where <code>dx</code> is a vector of points-to-plant x-distances, <code>dy</code> is a vector of points-to-plant y-distances, <code>marks</code> are the plant marks, and <code>par</code> receives the value of the <code>effpar</code> argument. Examples are provided in the functions <code>*.eff</code> ( <code>tass.eff</code> , etc.). The default is <code>flat.eff</code> , no weighting. |
| <code>effpar</code>     | Parameter(s) for efficiency, usually a list or vector.   |
| <code>plot</code>       | If <code>TRUE</code> , the denominator of the partition function is graphed as a pixel image, to visualize competition pressure (default).   |
| <code>afree</code>      | If <code>TRUE</code> , the free-growing assimilation is also computed. Default is <code>FALSE</code> .   |
| <code>centroid</code>   | If <code>TRUE</code> , the centroid of the plant assimilation distribution is also computed. Default is <code>FALSE</code> .   |

## 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 grid at a spatial resolution of `pixsize`, which is typically assumed to be uniform. Plants exert competitive pressure depending on size and distance, described by the `influence` function. The resource available at each pixel is allocated to plants according to their influence and to a partition rule parametrized by `partpar`. Finally, the resource uptake is weighted by the `efficiency` function, and is spatially integrated to obtain the plant's assimilation index.

## Value

Returns the point pattern `plants` with the results appended to the `marks(plants)` data frame. The additional marks are the assimilation indices in the 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

<http://forestgrowth.unbc.ca/siplab>

García, O. (2013) “A generic approach to spatial individual-based modelling and simulation of plant communities”. Manuscript in review.

## See Also

[influence](#), [efficiency](#), [edges](#)

## Examples

```
a <- assimilation(finpines, infpar=list(a=1, b=4,
    smark="height"), afree=TRUE)
head(marks(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 50 m × 60 m areas, 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 some trees with coordinates slightly 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

[http://daac.ornl.gov/cgi-bin/dsviewer.pl?ds\\_id=359](http://daac.ornl.gov/cgi-bin/dsviewer.pl?ds_id=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://www.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. <http://web.unbc.ca/~garcia/publ/distrib-CJFR.pdf>.

## Examples

```
data(boreasNP)
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 with replication.

## Usage

```
edges(plants, width)

core(plants, distance)
```

## Arguments

|          |  |
|----------|--|
| plants   | A <b>spatstat</b> point pattern object (class ppp). 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 into a torus. This function implements both strategies. When expanding, the part 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 result is 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. With a negative width, `edges(plants, width)` is the same as `plants[core(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 less than `width` from an edge of the enlarged 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` otherwise.

### Note

Requires the **spatstat** package.

### Author(s)

Oscar García.

### References

<http://forestgrowth.unbc.ca/siplab>

### See Also

[assimilation](#), [pairwise](#)

### Examples

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

**Description**

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

**Usage**

```
flat.eff(dx, dy, marks, par = NULL)
```

```
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. Points x-coordinates minus plant x-coordinate. |
| dy    | Vector of y-distances. Points y-coordinates minus plant y-coordinate. |
| marks | Plant mark information.   |
| par   | Vector or list of parameters.   |

**Details**

The values of par must be given in the argument effpar of [assimilation](#), they are shown here as examples.

smark in par indicates the location of the plant size variable in marks. It can be a data frame column number, or a string id like "height".

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**

<http://forestgrowth.unbc.ca/siplab>

García, O. "A generic approach to spatial individual-based modelling and simulation of plant communities". Manuscript in review.



**See Also**

[assimilation](#), [influence](#)

---

influence

---

*Influence Functions*


---

**Description**

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

**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. Points x-coordinates minus plant x-coordinate. |
| dy    | Vector of y-distances. Points y-coordinates minus plant y-coordinate. |
| marks | Plant mark information.   |
| par   | List of parameters.   |

**Details**

The values of par must be given in the argument `infpar` of [assimilation](#), they are shown here as examples.

`smark` in par indicates the location of the plant size variable in marks. It can be a data frame column number, or a string id like "height".

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

```
zoi.inf: 1 if  $R < kS$ , 0 otherwise
tass.inf:  $\max\{0, S - c[\exp(R/b) - 1]\}$ 
gates.inf:  $\max\{0, [(S/b)^a - R^a]^{1/a}\}$ 
gnomon.inf:  $\max\{0, S - bR^a\}$ 
```

**Value**

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

**Author(s)**

Oscar García.

**References**

<http://forestgrowth.unbc.ca/siplab>

García, O. "A generic approach to spatial individual-based modelling and simulation of plant communities". Manuscript in review.

**See Also**

[assimilation](#)

---

kernel

*Competition Kernel Functions*

---

**Description**

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

**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 must be given in the argument kerpar of pairwise, they are shown here as examples.

smark in par indicates the location of the plant size variable in marks. It can be a data frame column number, or a string id like "dbh".

Competition kernels seem to be limited only by the researchers imagination. 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, this kernel is  $(S_j^{p_j}/S_i^{p_i})/R^{p_r}$ . For instance, the popular Hegyi's index corresponds to  $p_i=1$ ,  $p_j=1$ ,  $p_r=1$ ,  $\text{smark}=\text{"dbh"}$ . This and other examples could be coded directly if computational efficiency is important.

`staebler.ker` is the width of the overlap of zones of influence (ZOI), used by Staebler in 1951. Assumes that the ZOI radius is related to size  $S$  by  $kS^p + c$ .

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

### Value

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

### Author(s)

Oscar García.

### References

<http://forestgrowth.unbc.ca/siplab>

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

### See Also

[pairwise](#)

---

pairwise

*Compute Pairwise Competition Indices*

---

### Description

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

### Usage

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

### Arguments

|                     |  |
|---------------------|--|
| <code>plants</code> | A <b>spatstat</b> point pattern object (class <code>ppp</code> ), containing the plants coordinates and marks with the plant size and possibly other attributes.                               |
| <code>maxN</code>   | Maximum number of nearest neighbors to include as potential competitors. Only one non-null <code>maxN</code> or <code>maxR</code> can be given. Default is <code>NULL</code> (no restriction). |
| <code>maxR</code>   | Maximum radius to search for potential competitors. Only one non-null <code>maxN</code> or <code>maxR</code> can be given. Default is <code>NULL</code> (no restriction).                      |

|                     |   |
|---------------------|---|
| <code>select</code> | Optional user-supplied selection function for choosing competitors. Must have arguments ( <code>imarks</code> , <code>jmarks</code> , <code>dists</code> , <code>dranks</code> , <code>par</code> ), where <code>imarks</code> are the marks for the subject plant (a 1-row data frame), <code>jmarks</code> is a data frame with the marks of the potential competitors, <code>dists</code> is a vector distances between subject plant and the potential competitors, <code>dranks</code> are the distance ranks, and <code>par</code> receives the value of the <code>selpar</code> argument. It must return a logical vector with the same length as <code>dists</code> . Examples are provided in the functions <code>*.sel</code> ( <a href="#">powlinear.sel</a> , etc.). Default is <code>NULL</code> (no selection). |
| <code>selpar</code> | Parameter(s) for <code>select</code> , usually a list or vector. Default: <code>NULL</code> .   |
| <code>kernel</code> | User-supplied competition kernel function for computing the effect of competitor $j$ on the subject plant $i$ . Must have arguments ( <code>imarks</code> , <code>jmarks</code> , <code>dists</code> , <code>dranks</code> , <code>par</code> ), where <code>imarks</code> are the marks for the subject plant (a 1-row data frame), <code>jmarks</code> is a data frame with the marks of the potential competitors, <code>dists</code> is a vector of distances between subject plant and the potential competitors, <code>dranks</code> are the distance ranks, and <code>par</code> receives the value of the <code>kerpar</code> argument. It must return a numeric vector with the same length as <code>dists</code> . Examples are provided in the functions <code>*.ker</code> ( <a href="#">powers.ker</a> , etc.).  |
| <code>kerpar</code> | Parameter(s) for <code>kernel</code> , usually a list or vector. Default: <code>NULL</code> .   |

## Details

Traditionally, a 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). A separate selection rule may reduce the computational effort of searching for neighbors.

Some simple selection rules can be implemented by giving a value to `maxN` or `maxR`. In any case, reasonable limits in one of these variables (they are mutually exclusive) may be advisable for reducing computation.

More complex rules can be specified by the `select` function, and parameters in `selpar`. See [select](#) for examples.

Kernel computation is specified by the `kernel` function, and parameters in `kerpar`. See [kernel](#) for examples.

## Value

Returns the point pattern plants, with the competition indices added to `marks(plants)` as a data frame column `cindex`.

## Note

Requires the **spatstat** package.

**Author(s)**

Oscar García.

**References**

<http://forestgrowth.unbc.ca/siplab>

**See Also**

[select](#), [kernel](#), [edges](#)

**Examples**

```
# Hegyi (1974) index
summary(pairwise(finpines, maxR = 6, kernel=powers.ker, kerpar =
  list(pi=1, pj=1, pr=1, r0=0.3048, smark="diameter")))
```

---

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](#).

**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` must be given in the argument `selpar` of `pairwise`, they are shown here as examples.

`smark` in `par` indicates the location of the plant size variable in `marks`. It can be a data frame column number, or a string id like "dbh".

`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) with particular values for the parameters (except for the *competition elimination angle*, which depends on relative positions among competitors):

Multiple of crown radius:  $k_j=0$ ,  $p=1$ ,  $r_0=0$ , `smark="crownwidth"`.

Angle count sampling:  $k_i=0$ ,  $p=1$ ,  $r_0=0$ , `smark="dbh"`.

Areas of influence overlap:  $k_i=k_j$ ,  $p=1$ ,  $r_0=0$ , if the radius is a linear function of size ( $p$  not 1 for an allometric relationship).

Vertical search cone:  $k_i = -c_i / \tan(90 - \beta/2)$ ,  $k_j = 1 / \tan(90 - \beta/2)$ ,  $p=1$ , `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

<http://forestgrowth.unbc.ca/siplab>

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

## See Also

`pairwise`

# Index

## \*Topic **datasets**

boreas, [5](#)

assimilation, [2](#), [3](#), [7–10](#)

boreas, [5](#)

boreasNP, [3](#)

boreasNP (boreas), [5](#)

boreasNS, [3](#)

boreasNS (boreas), [5](#)

boreasSA, [3](#)

boreasSA (boreas), [5](#)

boreasSP, [3](#)

boreasSP (boreas), [5](#)

core (edges), [6](#)

edges, [2](#), [5](#), [6](#), [13](#)

efficiency, [2](#), [4](#), [5](#), [8](#)

flat.eff, [4](#)

flat.eff (efficiency), [8](#)

gates.eff (efficiency), [8](#)

gates.inf (influence), [9](#)

gnomon.eff (efficiency), [8](#)

gnomon.inf, [3](#)

gnomon.inf (influence), [9](#)

influence, [2](#), [4](#), [5](#), [8](#), [9](#), [9](#)

kernel, [10](#), [12](#), [13](#)

pairwise, [2](#), [7](#), [10](#), [11](#), [11](#), [13](#), [14](#)

powers.ker, [12](#)

powers.ker (kernel), [10](#)

powlinear.sel, [12](#)

powlinear.sel (select), [13](#)

select, [12](#), [13](#), [13](#)

siplab (siplab-package), [2](#)

siplab-package, [2](#)

size.sel (select), [13](#)

spurr.ker (kernel), [10](#)

staebler.ker (kernel), [10](#)

tass.eff, [4](#)

tass.eff (efficiency), [8](#)

tass.inf, [3](#)

tass.inf (influence), [9](#)

zoi.inf (influence), [9](#)