Package: spatialcluster (via r-universe)

November 10, 2026

Title Spatially Contiguous Clustering with the 'Redcap' Algorithm

Version 0.2.1.009

Description Groups spatial data into clusters of similar observations, while requiring that each cluster forms a single spatially contiguous region. Contiguity is enforced by first building a spanning tree from the spatial relationships between points, and then cutting that tree into clusters according to a matrix of distances or other measures of similarity. Trees may be built from first-order relationships only (direct neighbours), yielding a minimum spanning tree, or from full-order relationships between entire clusters, using single linkage (nearest members), average linkage (mean of all pairs), or complete linkage (furthest members). The single and average linkage methods implement the 'Redcap' algorithm ('Regionalization' with dynamically constrained agglomerative clustering and partitioning). Results can be visualised as maps of clusters, cluster hulls, or trees.

License GPL-3

URL https://UrbanAnalyst.github.io/spatialcluster/, https://github.com/UrbanAnalyst/spatialcluster

BugReports https://github.com/UrbanAnalyst/spatialcluster/issues

Depends R (>= 4.1.0)

Imports alphahull, deldir, dplyr, ggplot2, ggthemes, methods, Rcpp (>= 0.12.6), tibble

Suggests cdtr, dbscan, knitr, rmarkdown, roxygen2, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation yes

Config/roxygen2/version 8.1.1

Repository https://urbananalyst.r-universe.dev

Date/Publication 2026-11-06 10:08:17 UTC

RemoteUrl https://github.com/UrbanAnalyst/spatialcluster

RemoteRef HEAD

RemoteSha 5e3920df852cc959c95b72e173dab2ab0cd1c1dc

Contents

plot.scl 2
plot_merges 3
scl_full 4
scl_recluster 5
scl_redcap 6
spatialcluster 8

Index **10**

`plot.scl` *plot.scl*

Description

plot.scl

Usage

```
## S3 method for class 'scl'
plot(x, ..., hull_alpha = 1)
```

Arguments

`x`	object to be plotted
`...`	ignored here
`hull_alpha`	alpha value of (non-)convex hulls, with default generating a convex hull, and smaller values generating concave hulls. (See ?alphashape::ashape for details).

Value

A **ggplot2** object.

See Also

Other plot_fns: `plot_merges()`

Examples

```
set.seed (1)
n <- 100
xy <- matrix (runif (2 * n), ncol = 2)
dmat <- matrix (runif (n^2), ncol = n)
scl <- scl_redcap (xy, dmat, ncl = 4)
plot (scl)
# Connect clusters according to highest (\code{shortest = FALSE}) values of
# \code{dmat}:
scl <- scl_redcap (xy, dmat, ncl = 4, shortest = FALSE, full_order = FALSE)
plot (scl)
```

`plot_merges` *plot_merges*

Description

Plot dendrogram of merges for `scl` object with `method = "full"`.

Usage

```
plot_merges(x, root_tree = FALSE)
```

Arguments

`x`	Object of class `scl` obtained with `method = "full"`.
`root_tree`	If `TRUE`, tree leaves are connected to bottom of plot, otherwise floating as determined by plot.hclust.

Value

Nothing (generates plot)

See Also

Other plot_fns: `plot.scl()`

Examples

```
n <- 100
xy <- matrix (runif (2 * n), ncol = 2)
dmat <- matrix (runif (n^2), ncol = n)
scl <- scl_full (xy, dmat, ncl = 4)
plot_merges (scl)
```

scl_full *scl_full*

Description

Full spatially-constrained clustering.

Usage

```
scl_full(
  xy,
  dmat,
  ncl,
  linkage = "single",
  shortest = TRUE,
  nnbs = 6L,
  boundary = NULL
)
```

Arguments

xy	Rectangular structure (matrix, data.frame, tibble), containing coordinates of points to be clustered.
dmat	Structure (matrix, data.frame, tibble) containing distances or equivalent metrics between all points in xy. If xy has n rows, then dmat may either be square, with n rows and n columns, or in long form, with three columns of (from, to, value). In long form, from and to are either integer row indices of xy or character values matching rownames (xy). Values may be given for one direction of each pair only, and are then mirrored; values for all pairs must be present.
ncl	Desired number of clusters. See description of 'ncl_iterate' parameter for conditions under which actual number may be less than this value.
linkage	One of "single", "average", or "complete". For covariance clustering, use shortest = FALSE; "complete" linkage then uses the smallest connecting value between clusters. As for scl_redcap, initial clusters are always formed from spatially nearest neighbours; shortest and linkage determine only how these are then merged according to dmat.
shortest	If TRUE, the dmat is interpreted as distances such that lower values indicate stronger relationships, and the tree is cut to minimise the within-cluster sum of squared deviations of dmat values. If FALSE, then higher values of dmat are interpreted to indicate stronger relationships, as is the case for example with covariances, and the tree is cut to maximise the mean dmat value within clusters. This parameter only affects how the tree is cut; it has no effect on construction of the spanning tree, which is always based on spatial distances.
nnbs	Number of spatially nearest neighbours to be used in calculating clustering trees, irrespective of shortest. Triangulation will be used if nnbs <= 0. Ignored if boundary is given.

boundary — Optional boundary of the study region. If given, spatial neighbours are taken from a constrained Delaunay triangulation of the points within this boundary (requires the 'cdtr' package), so that no neighbour edges cross outside it, for example across concavities in the region. May be a two-column matrix or data frame of the vertices of a single ring, a list of such rings (holes are recognised by nesting), or an 'sf' polygon or multipolygon object. All points in xy must lie inside the boundary.

Value

A object of class scl with tree containing the clustering scheme, and xy the original coordinate data of the clustered points. An additional component, tree_rest, enables the tree to be re-cut to a different number of clusters via scl_recluster.

See Also

Other clustering_fns: scl_recluster(), scl_redcap()

Examples

```
n <- 100
xy <- matrix (runif (2 * n), ncol = 2)
dmat <- matrix (runif (n^2), ncol = n)
scl <- scl_full (xy, dmat, ncl = 4)
```

---

scl_recluster *scl_recluster*

Description

Re-cut a spatial cluster tree (scl) at a different number of clusters.

Usage

```
scl_recluster(scl, ncl, shortest = TRUE, quiet = FALSE)
```

Arguments

scl — An scl object returned from scl_redcap.

ncl — Desired number of clusters. See description of 'ncl_iterate' parameter for conditions under which actual number may be less than this value.

shortest — If TRUE, the dmat is interpreted as distances such that lower values indicate stronger relationships, and the tree is cut to minimise the within-cluster sum of squared deviations of dmat values. If FALSE, then higher values of dmat are interpreted to indicate stronger relationships, as is the case for example with covariances, and the tree is cut to maximise the mean dmat value within clusters. This parameter only affects how the tree is cut; it has no effect on construction of the spanning tree, which is always based on spatial distances.

quiet — If 'FALSE' (default), display progress information on screen.

Value

Modified `scl` object in which `tree` is re-cut into `ncl` clusters.

See Also

Other clustering_fns: `scl_full()`, `scl_redcap()`

Examples

```
n <- 100
xy <- matrix (runif (2 * n), ncol = 2)
dmat <- matrix (runif (n^2), ncol = n)
scl <- scl_redcap (xy, dmat, ncl = 4)
plot (scl)
scl <- scl_recluster (scl, ncl = 5)
plot (scl)
```

`scl_redcap` *scl_redcap*

Description

Cluster spatial data with REDCAP (REgionalization with Dynamically Constrained Agglomerative clustering and Partitioning) routines.

Usage

```
scl_redcap(
  xy,
  dmat,
  ncl,
  full_order = TRUE,
  linkage = "single",
  shortest = TRUE,
  nnbs = 6L,
  boundary = NULL,
  iterate_ncl = FALSE,
  quiet = FALSE
)
```

Arguments

`xy`	Rectangular structure (matrix, data.frame, tibble), containing coordinates of points to be clustered.

dmat	Structure (matrix, data.frame, tibble) containing distances or equivalent metrics between all points in xy. If xy has n rows, then dmat may either be square, with n rows and n columns, or in long form, with three columns of (from, to, value). In long form, from and to are either integer row indices of xy or character values matching rownames(xy). Values may be given for one direction of each pair only, and are then mirrored; values for all pairs must be present.
ncl	Desired number of clusters. See description of 'ncl_iterate' parameter for conditions under which actual number may be less than this value.
full_order	If FALSE, build spanning trees from first-order relationships only, otherwise build from full-order relationships (see Note).
linkage	One of "single", "average", or "complete"; see Note.
shortest	If TRUE, the dmat is interpreted as distances such that lower values indicate stronger relationships, and the tree is cut to minimise the within-cluster sum of squared deviations of dmat values. If FALSE, then higher values of dmat are interpreted to indicate stronger relationships, as is the case for example with covariances, and the tree is cut to maximise the mean dmat value within clusters. This parameter only affects how the tree is cut; it has no effect on construction of the spanning tree, which is always based on spatial distances.
nnbs	Number of spatially nearest neighbours to be used in calculating clustering trees, irrespective of shortest. Triangulation will be used if nnbs <= 0. Ignored if boundary is given.
boundary	Optional boundary of the study region. If given, spatial neighbours are taken from a constrained Delaunay triangulation of the points within this boundary (requires the 'cdtr' package), so that no neighbour edges cross outside it, for example across concavities in the region. May be a two-column matrix or data frame of the vertices of a single ring, a list of such rings (holes are recognised by nesting), or an 'sf' polygon or multipolygon object. All points in xy must lie inside the boundary.
iterate_ncl	Actual numbers of clusters found may be less than the specified value of 'ncl', because clusters formed from < 3 edges are removed. If 'iterate_ncl = FALSE' (the default), the value is returned with whatever number of actual clusters is found. Setting this parameter to 'TRUE' forces the algorithm to iterate until the exact number of clusters has been found. For large data sets, this may result in considerable longer calculation times.
quiet	If 'FALSE' (default), display progress information on screen.

Details

Clustering is in two stages. First, a spanning tree is constructed from the **spatial** relationships between points in xy alone, using the nearest spatial neighbours (see nnbs), and independent of both dmat and shortest. Second, that tree is cut into ncl clusters by evaluating the values of dmat along its edges.

Value

A object of class scl with tree containing the clustering scheme, and xy the original coordinate data of the clustered points. An additional component, tree_rest, enables the tree to be re-cut to a different number of clusters via scl_recluster, rather than calculating clusters anew.

Note

Please refer to the original REDCAP paper ('Regionalization with dynamically constrained agglomerative clustering and partitioning (REDCAP)', by D. Guo (2008), Int.J.Geo.Inf.Sci 22:801-823) for details of the `full_order` and `linkage` parameters. Note that, unlike the original, the spanning trees constructed here (for all values of `linkage`) are based on spatial distances between points in `xy`, with `dmat` used only when cutting the trees into clusters. This paper clearly demonstrates the general inferiority of spanning trees constructed from first-order relationships. It is therefore strongly recommended that the default `full_order = TRUE` be used at all times.

See Also

Other clustering_fns: `scl_full()`, `scl_recluster()`

Examples

```
n <- 100
xy <- matrix (runif (2 * n), ncol = 2)
dmat <- matrix (runif (n^2), ncol = n)
scl <- scl_redcap (xy, dmat, ncl = 4)
# By default, the spatial tree is cut to minimise within-cluster variation
# of (\code{shortest}) values of \code{dmat}. Setting \code{shortest = FALSE}
# instead cuts the same tree to maximise mean within-cluster values of
# \code{dmat}, as for covariances, and will generally give different
# clusters:
scl <- scl_redcap (xy, dmat, ncl = 4, shortest = FALSE)
# using 'full_order = FALSE' constructs clusters from first-order
# relationships only; not recommended, but possible nevertheless:
scl <- scl_redcap (xy, dmat, ncl = 4, full_order = FALSE)
```

`spatialcluster` *Spatially Contiguous Clustering with the 'Redcap' Algorithm*

Description

Description: Groups spatial data into clusters of similar observations, while requiring that each cluster forms a single spatially contiguous region. Contiguity is enforced by first building a spanning tree from the spatial relationships between points, and then cutting that tree into clusters according to a matrix of distances or other measures of similarity. Trees may be built from first-order relationships only (direct neighbours), yielding a minimum spanning tree, or from full-order relationships between entire clusters, using single linkage (nearest members), average linkage (mean of all pairs), or complete linkage (furthest members). The single and average linkage methods implement the 'Redcap' algorithm ('Regionalization' with dynamically constrained agglomerative clustering and partitioning). Results can be visualised as maps of clusters, cluster hulls, or trees.

Author(s)

Maintainer: Mark Padgham <mark.padgham@email.com> (ORCID)

Authors:

- Mark Padgham <mark.padgham@email.com> (ORCID)

Other contributors:

- Michael Sumner (ORCID) [contributor]

See Also

Useful links:

- https://UrbanAnalyst.github.io/spatialcluster/
- https://github.com/UrbanAnalyst/spatialcluster
- Report bugs at https://github.com/UrbanAnalyst/spatialcluster/issues

Index

∗ **clustering_fns**
    scl_full, 4
    scl_recluster, 5
    scl_redcap, 6
∗ **package**
    spatialcluster, 8
∗ **plot_fns**
    plot.scl, 2
    plot_merges, 3

plot.hclust, *3*
plot.scl, 2
plot.scl(), *3*
plot_merges, 3
plot_merges(), *2*

scl_full, 4
scl_full(), *6*, *8*
scl_recluster, 5, *5*, *7*
scl_recluster(), *5*, *8*
scl_redcap, *4*, *5*, 6
scl_redcap(), *5*, *6*
spatialcluster, 8
spatialcluster-package
        *(*spatialcluster*)*, 8