# Package: supercells (via r-universe)

August 30, 2024

**Title** Superpixels of Spatial Data

**Version** 1.0.3

**Description** Creates superpixels based on input spatial data. This package works on spatial data with one variable (e.g., continuous raster), many variables (e.g., RGB rasters), and spatial patterns (e.g., areas in categorical rasters). It is based on the SLIC algorithm (Achanta et al. (2012) <doi:10.1109/TPAMI.2012.120>), and readapts it to work with arbitrary dissimilarity measures.

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** sf, terra (>= 1.4-21), philentropy (>= 0.6.0), future.apply

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**LinkingTo** cpp11

**URL** https://jakubnowosad.com/supercells/

**BugReports** https://github.com/Nowosad/supercells/issues

**Suggests** knitr, covr, testthat (>= 3.0.0), rmarkdown, stars

**Config/testthat/edition** 3

**Repository** https://nowosad.r-universe.dev

**RemoteUrl** https://github.com/nowosad/supercells

**RemoteRef** HEAD

**RemoteSha** 394d1a4c983a0a062379fabf35dfb7d61ab4ca6b

## Contents

`supercells` *Creates supercells*

## Description

Creates supercells based on single- or multi-band spatial raster data. It uses a modified version of the SLIC Superpixel algorithm by Achanta et al. (2012), allowing specification of a distance function.

## Usage

```
supercells(
  x,
  k,
  compactness,
  dist_fun = "euclidean",
  avg_fun = "mean",
  clean = TRUE,
  iter = 10,
  transform = NULL,
  step,
  minarea,
  metadata = TRUE,
  chunks = FALSE,
  future = FALSE,
  verbose = 0
)
```

## Arguments

| | |
|---|---|
| `x` | An object of class SpatRaster (terra) or class stars (stars) |
| `k` | The number of supercells desired by the user (the output number can be slightly different!). You can use either `k` or `step`. It is also possible to provide a set of points (an `sf` object) as `k` together with the `step` value to create custom cluster centers. |
| `compactness` | A compactness value. Larger values cause clusters to be more compact/even (square). A compactness value depends on the range of input cell values and selected distance measure. |
| `dist_fun` | A distance function. Currently implemented distance functions are `"euclidean"`, `"jsd"`, `"dtw"` (dynamic time warping), name of any distance function from the `philentropy` package (see `philentropy::getDistMethods()`; "log2" is used in this case), or any user defined function accepting two vectors and returning one value. Default: `"euclidean"` |
| `avg_fun` | An averaging function - how the values of the supercells' centers are calculated? The algorithm internally implements common functions `"mean"` and `"median"` (provided with quotation marks), but also accepts any fitting R function (e.g., |

| | |
|---|---|
| | base::mean() or stats::median(), provided as plain function name: mean). Default: "mean". See details for more information. |
| clean | Should connectivity of the supercells be enforced? |
| iter | The number of iterations performed to create the output. |
| transform | Transformation to be performed on the input. By default, no transformation is performed. Currently available transformation is "to_LAB": first, the conversion from RGB to the LAB color space is applied, then the supercells algorithm is run, and afterward, a reverse transformation is performed on the obtained results. (This argument is experimental and may be removed in the future). |
| step | The distance (number of cells) between initial supercells' centers. You can use either k or step. |
| minarea | Specifies the minimal size of a supercell (in cells). Only works when clean = TRUE. By default, when clean = TRUE, average area (A) is calculated based on the total number of cells divided by a number of supercells Next, the minimal size of a supercell equals to A/(2^2) (A is being right shifted) |
| metadata | Logical. If TRUE, the output object will have metadata columns ("supercells", "x", "y"). If FALSE, the output object will not have metadata columns. |
| chunks | Should the input (x) be split into chunks before deriving supercells? Either FALSE (default), TRUE (only large input objects are split), or a numeric value (representing the side length of the chunk in the number of cells). |
| future | Should the future package be used for parallelization of the calculations? Default: FALSE. If TRUE, you also need to specify future::plan(). |
| verbose | An integer specifying the level of text messages printed during calculations. 0 means no messages (default), 1 provides basic messages (e.g., calculation stage). |

## Details

If you want to use additional arguments for the averaging function (avg_fun), you can create a custom function. For example, if you want to calculate the mean by removing missing values, you can use the following code: my_mean = function(x) mean(x, na.rm = TRUE) and then provide avg_fun = my_mean.

## Value

An sf object with several columns: (1) supercells - an id of each supercell, (2) y and x coordinates, (3) one or more columns with average values of given variables in each supercell

## References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11), 2274–2282. https://doi.org/10.1109/tpami.2012.120

Nowosad, J. Motif: an open-source R tool for pattern-based spatial analysis. Landscape Ecol (2021). https://doi.org/10.1007/s10980-020-01135-0

## Examples

```
library(supercells)
# One variable

vol = terra::rast(system.file("raster/volcano.tif", package = "supercells"))
vol_slic1 = supercells(vol, k = 50, compactness = 1)
terra::plot(vol)
plot(sf::st_geometry(vol_slic1), add = TRUE, lwd = 0.2)

# RGB variables
# ortho = terra::rast(system.file("raster/ortho.tif", package = "supercells"))
# ortho_slic1 = supercells(ortho, k = 1000, compactness = 10, transform = "to_LAB")
# terra::plot(ortho)
# plot(sf::st_geometry(ortho_slic1), add = TRUE)
#
# ### RGB variables - colored output
#
# rgb_to_hex = function(x){
#   apply(t(x), 2, function(x) rgb(x[1], x[2], x[3], maxColorValue = 255))
# }
# avg_colors = rgb_to_hex(sf::st_drop_geometry(ortho_slic1[4:6]))
#
# terra::plot(ortho)
# plot(sf::st_geometry(ortho_slic1), add = TRUE, col = avg_colors)
```

# Index