Package: snic (via r-universe)

May 13, 2026

Title Superpixel Segmentation with the Simple Non-Iterative Clustering Algorithm

Version 0.6.1

Maintainer Rolf Simoes <rolfsimoes@gmail.com>

Description Implements the Simple Non-Iterative Clustering algorithm for superpixel segmentation of multi-band images, as introduced by Achanta and Susstrunk (2017) <doi:10.1109/CVPR.2017.520>. Supports both standard image arrays and geospatial raster objects, with a design that can be extended to other spatial data frameworks. The algorithm groups adjacent pixels into compact, coherent regions based on spectral similarity and spatial proximity. A high-performance implementation supports images with arbitrary spectral bands.

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), terra, spelling, covr, magick, jpeg, png, knitr, rmarkdown

URL https://github.com/rolfsimoes/snic, https://rolfsimoes.github.io/snic/

BugReports https://github.com/rolfsimoes/snic/issues

ByteCompile true

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

Encoding UTF-8

Language en-US

License GPL (>= 2)

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

Date/Publication 2025-12-14 03:12:37 UTC

RemoteUrl https://github.com/rolfsimoes/snic

RemoteRef HEAD

RemoteSha 102f0b6d69c2328c1daaea2931664fe6aad9f312

Contents

`seeds_api` *Convert seed coordinates between raster index, map, and WGS84 systems*

Description

These helpers make it easy to express an existing set of seed coordinates in a different coordinate system. Seeds are accepted when they already contain the target column pair and otherwise are projected using the provided raster.

Usage

```
as_seeds_rc(seeds, x)

as_seeds_xy(seeds, x)

as_seeds_wgs84(seeds, x)
```

Arguments

`seeds`	A data frame, `tibble`, or matrix that contains one of the column pairs `(r, c)`, `(x, y)`, or `(lat, lon)`.
`x`	A `SpatRaster` or in-memory array that supplies the metadata needed to perform conversions. For array inputs, note that the internal coordinate system has its y-axis inverted (increasing upwards) compared to matrix indices (which increase downwards).

Value

A data frame with the target coordinate columns and any additional columns that were present in `seeds`.

Target systems

`as_seeds_rc()` Ensures seeds are expressed in raster row / column indices (`r`, `c`).

`as_seeds_xy()` Ensures seeds use the raster CRS in map units (`x`, `y`).

`as_seeds_wgs84()` Ensures seeds are in geographic coordinates (`lat`, `lon`) in `EPSG:4326`.

Note

Only rasters with a defined coordinate reference system (CRS) can be transformed to WGS84 coordinates. If the input raster ‘x‘ lacks a CRS, attempting to convert to WGS84 will result in an error.

Examples

```
if (requireNamespace("terra", quietly = TRUE) && terra_is_working()) {
    # Load a test Sentinel-2 band
    s2_file <- system.file(
        "demo-geotiff/S2_20LMR_B04_20220630.tif",
        package = "snic"
    )
    s2_rast <- terra::rast(s2_file)

    # Create some test coordinates in pixel space
    seeds_rc <- data.frame(r = c(10, 20, 30), c = c(15, 25, 35))

    # Convert to map coordinates (x,y)
    seeds_xy <- as_seeds_xy(seeds_rc, s2_rast)

    # Convert to geographic coordinates (lat,lon)
    seeds_wgs84 <- as_seeds_wgs84(seeds_rc, s2_rast)
}
```

`snic` *Simple Non-Iterative Clustering (SNIC) segmentation*

Description

Segment an image into superpixels using the SNIC algorithm. This function wraps a C++ implementation operating on any number of spectral bands and uses 4-neighbor (von Neumann) connectivity.

Usage

```
snic(x, seeds, compactness = 0.5, ...)
```

Arguments

x	Image data. For the `array` method this must be a numeric array with dimensions `(height, width, bands)` in column-major order (R's native storage). For the `SpatRaster` method (from **terra**), dimensions and band ordering are inferred automatically.
seeds	Initial seed coordinates. The required format depends on the spatial status of `x`: • If `x` has no CRS: a two-column data frame `(r, c)` giving 1-based pixel coordinates. • If `x` has a CRS: a two-column data frame with columns `lat` and `lon` expressed in `EPSG:4326`. These are converted internally to pixel coordinates before segmentation. Seeds define the starting cluster centers. They are usually generated with `snic_grid` helpers (e.g. rectangular, hexagonal or random), or placed interactively via `snic_grid_manual`.
compactness	Non-negative numeric value controlling the balance between feature similarity and spatial proximity (default = 0.5). Larger values produce more spatially compact superpixels.
...	Currently unused; reserved for future extensions.

Details

The algorithm performs clustering in a joint space that includes the image's spectral dimensions and two spatial coordinates. Each seed initializes a region, and pixels are assigned based on the SNIC distance metric combining spectral similarity and spatial distance, weighted by `compactness`.

Value

An object of class `snic` bundling the segmentation result together with per-cluster summaries produced by the SNIC algorithm. The segmentation result can be accessed using `snic_get_seg`. The per-cluster summaries can be accessed using `snic_get_means` and `snic_get_centroids`.

See Also

`snic_grid` for seed generation, `snic_grid_manual` for interactive placement, `snic_plot` for visualizing results.

Examples

```
# Example 1: Geospatial raster
if (requireNamespace("terra", quietly = TRUE) && terra_is_working()) {
    path <- system.file("demo-geotiff", package = "snic", mustWork = TRUE)
    files <- file.path(
        path,
        c(
            "S2_20LMR_B02_20220630.tif",
            "S2_20LMR_B04_20220630.tif",
            "S2_20LMR_B08_20220630.tif",
            "S2_20LMR_B12_20220630.tif"
```

```
        )
    )

    # Downsample for speed (optional)
    s2 <- terra::aggregate(terra::rast(files), fact = 8)

    # Generate a regular grid of seeds (lat/lon because CRS is present)
    seeds <- snic_grid(
        s2,
        type    = "rectangular",
        spacing = 10L,
        padding = 18L
    )

    # Run segmentation
    seg <- snic(s2, seeds, compactness = 0.25)

    # Visualize RGB composite with seeds and segment boundaries
    snic_plot(
        s2,
        r = 4, g = 3, b = 1,
        stretch = "lin",
        seeds = seeds,
        seg = seg
    )
}

# Example 2: In-memory image (JPEG) + Lab transform
# Uses an example image shipped with the package (no terra needed)
if (requireNamespace("jpeg", quietly = TRUE)) {
    img_path <- system.file(
        "demo-jpeg/clownfish.jpeg",
        package = "snic",
        mustWork = TRUE
    )
    rgb <- jpeg::readJPEG(img_path) # h x w x 3 in [0, 1]

    # Convert sRGB -> CIE Lab for perceptual clustering
    dims <- dim(rgb)
    dim(rgb) <- c(dims[1] * dims[2], dims[3])
    lab <- grDevices::convertColor(
        rgb,
        from = "sRGB",
        to = "Lab",
        scale.in = 1,
        scale.out = 1 / 255
    )
    dim(lab) <- dims
    dim(rgb) <- dims

    # Seeds in pixel coordinates for array inputs
    seeds_rc <- snic_grid(lab, type = "hexagonal", spacing = 20L)
```

```
    # Segment in Lab space and plot L channel with boundaries
    seg <- snic(lab, seeds_rc, compactness = 0.1)

    snic_plot(
        rgb,
        r = 1L,
        g = 2L,
        b = 3L,
        seg = seg,
        seg_plot_args = list(
            border = "black"
        )
    )
}
```

`snic_animation` *Animated visualization of SNIC seeding and segmentation*

Description

Generate an animated GIF illustrating how SNIC segmentation evolves as seeds are progressively added. This function runs a sequence of SNIC segmentations using incremental subsets of the provided seeds and compiles the results into an animation.

Usage

```
snic_animation(
  x,
  seeds,
  file_path,
  max_frames = 100L,
  delay = 10,
  progress = getOption("snic.progress", FALSE),
  ...,
  snic_args = list(compactness = 0.5),
  device_args = list(res = 96, bg = "white")
)
```

Arguments

`x`	A `SpatRaster` representing the image to segment. Dimensions and coordinate reference are inferred automatically.
`seeds`	A two-column object specifying seed coordinates. If `x` has a CRS, use columns `lat` and `lon` (in `EPSG:4326`); otherwise use pixel indices (`r`, `c`). Typically created with `snic_grid` or interactively with `snic_grid_manual`.
`file_path`	Path where the resulting GIF is saved. The file must not already exist and the parent directory must be writable.

max_frames	Maximum number of frames to render. If there are more seeds than max_frames, only the first max_frames seeds are used.
delay	Per-frame delay in centiseconds (1/100 s). Passed to magick::image_animate(). Default is 10 (0.1 s per frame).
progress	Logical scalar; if TRUE, show the textual progress bar while generating frames.
...	Additional arguments forwarded to snic_plot when drawing each frame (e.g., RGB band indices or palette options).
snic_args	Named list of extra arguments passed to snic on every iteration (e.g., compactness). Arguments x and seeds are reserved and cannot be overridden.
device_args	Named list of arguments passed to grDevices::png() when rendering frames. Defaults to list(res = 96, bg = "white"). Values such as width, height, and filename are managed automatically.

Details

For each iteration, the function adds one seed to the current set and re-runs snic. The segmentation and seed locations are drawn using snic_plot, saved as PNGs, and then combined into an animated GIF using the **magick** package. This is intended for exploratory and didactic use to illustrate the influence of seed placement and parameters such as compactness.

Value

Invisibly, the file path of the generated GIF.

See Also

snic, snic_plot, snic_grid, snic_grid_manual.

Examples

```
if (requireNamespace("terra", quietly = TRUE) &&
    terra_is_working() &&
    requireNamespace("magick", quietly = TRUE)) {
    tif_dir <- system.file("demo-geotiff", package = "snic", mustWork = TRUE)
    files <- file.path(
        tif_dir,
        c(
            "S2_20LMR_B02_20220630.tif",
            "S2_20LMR_B04_20220630.tif",
            "S2_20LMR_B08_20220630.tif",
            "S2_20LMR_B12_20220630.tif"
        )
    )
    s2 <- terra::aggregate(terra::rast(files), fact = 8)

    set.seed(42)
    seeds <- snic_grid(s2, type = "random", spacing = 10L, padding = 0L)

    gif_file <- snic_animation(
        s2,
```

```
        seeds = seeds,
        file_path = tempfile("snic-demo", fileext = ".gif"),
        max_frames = 20L,
        snic_args = list(compactness = 0.1),
        r = 4, g = 3, b = 1,
        device_args = list(height = 192, width = 256)
    )
    gif_file
}
```

`snic_class` *SNIC segmentation container*

Description

Objects returned by `snic` inherit from the `snic` S3 class. They are lightweight containers bundling the segmentation result together with per-cluster summaries produced by the SNIC algorithm. Internally, a `snic` object is a named list with components:

`seg` The segmentation map in the native type of the input (either a 3D integer `array` with dimensions `(height, width, 1)` or a single-layer `SpatRaster`—matching the input given to `snic()`). Values are superpixel labels (positive integers); `NA` marks pixels that were not assigned.

`means` Numeric matrix with one row per superpixel and one column per input band (column names preserved when available), giving the mean feature value of each cluster. May be `NULL` if the backend cannot retain these summaries.

`centroids` Numeric matrix with columns `r` and `c` giving the cluster centers in pixel coordinates (0-based indices used by the SNIC core). May be `NULL` when centroids are unavailable.

The list carries class `"snic"` to enable the accessors and print methods below; the segmentation labels index the corresponding rows of `means` and `centroids`.

Usage

```
snic_get_means(x)

snic_get_centroids(x)

snic_get_seg(x)

## S3 method for class 'snic'
snic_get_means(x)

## S3 method for class 'snic'
snic_get_centroids(x)

## S3 method for class 'snic'
snic_get_seg(x)
```

```
## S3 method for class 'snic'
print(x, ...)
```

Arguments

x	A snic object, typically the result of a call to snic. It stores the segmentation map along with per-cluster summaries (means, centroids, and metadata) produced by the SNIC algorithm.
...	Additional arguments passed to or from methods. Currently unused, but included for compatibility with S3 method dispatch.

Value

- snic_get_means(): Numeric matrix of per-cluster band means, or NULL when not available.
- snic_get_centroids(): Numeric matrix with columns r and c giving cluster centers in pixel coordinates (0-based), or NULL when not available.
- snic_get_seg(): Segmentation map in the native type of the input (array or SpatRaster) with integer labels and possible NA for unassigned pixels.
- print.snic(): Invisibly returns x after printing a human-readable summary.

Accessors

- snic_get_seg: Retrieve the segmentation result.
- snic_get_means: Retrieve per-cluster feature means.
- snic_get_centroids: Retrieve per-cluster centroids.

Methods

- snic_animation: Animate the segmentation process.
- print: Print a summary of the segmentation result.
- plot: Visualize the segmentation result.

---

snic_grid *Spatial grid seeding for SNIC segmentation*

Description

Generate seed locations on an image following one of four spatial arrangements used in SNIC (Simple Non-Iterative Clustering) segmentation: rectangular, diamond, hexagonal, or random. Works for both numeric arrays and SpatRaster objects.

Usage

```
snic_grid(
  x,
  type = c("rectangular", "diamond", "hexagonal", "random"),
  spacing,
  padding = spacing/2,
  ...
)

snic_count_seeds(x, spacing, padding = spacing/2)
```

Arguments

x	Image data. For arrays, this must be numeric with dimensions (height, width, bands). For SpatRaster objects, raster dimensions are inferred automatically.
type	Character string indicating the spatial pattern to generate. One of "rectangular", "diamond", "hexagonal", or "random".
spacing	Numeric or integer. Either one value (applied to both axes) or two values (vertical, horizontal) giving the spacing between seeds in pixels.
padding	Numeric or integer. Distance from image borders within which no seeds are placed. May be of length 1 or 2. Defaults to spacing / 2.
...	Currently unused; reserved for future extensions.

Details

The spacing parameter controls seed density. Padding shifts the seed grid inward so that seeds are not placed directly on image borders.

The spatial arrangements are:

- rectangular: regular grid aligned with rows and columns.
- diamond: alternating row offsets, forming a diamond layout.
- hexagonal: alternating offsets approximating a hexagonal tiling.
- random: uniform random placement with similar expected density.

The helper snic_count_seeds estimates how many seeds would be generated for a rectangular lattice with the given spacing and padding, without computing coordinates. For type = "diamond" or "hexagonal", the actual number of seeds will be up to roughly twice this estimate (minus boundary effects). For "random", the estimate corresponds to the expected density.

If x has a coordinate reference system, the returned data frame includes geographic coordinates (lat, lon) in EPSG:4326.

Value

A data frame containing:

- r, c when x has no CRS.
- lat, lon when x has a CRS, expressed in EPSG:4326.

See Also

snic_count_seeds for estimating seed counts.

Examples

```
# Example 1: Geospatial raster
if (requireNamespace("terra", quietly = TRUE) && terra_is_working()) {
    # Load example multi-band image (Sentinel-2 subset) and downsample
    tiff_dir <- system.file("demo-geotiff",
        package = "snic",
        mustWork = TRUE
    )
    files <- file.path(tiff_dir, c(
        "S2_20LMR_B02_20220630.tif",
        "S2_20LMR_B04_20220630.tif",
        "S2_20LMR_B08_20220630.tif",
        "S2_20LMR_B12_20220630.tif"
    ))
    s2 <- terra::aggregate(terra::rast(files), fact = 8)

    # Compare grid types visually using snic_plot for immediate feedback
    types <- c("rectangular", "diamond", "hexagonal", "random")
    op <- par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
    for (tp in types) {
        seeds <- snic_grid(s2, type = tp, spacing = 12L, padding = 18L)
        snic_plot(
            s2,
            r = 4, g = 3, b = 1, stretch = "lin",
            seeds = seeds,
            main = paste("Grid:", tp)
        )
    }
    par(mfrow = c(1, 1))

    # Estimate seed counts for planning
    snic_count_seeds(s2, spacing = 12L, padding = 18L)
    par(op)
}

# Example 2: In-memory image (JPEG)
if (requireNamespace("jpeg", quietly = TRUE)) {
    img_path <- system.file(
        "demo-jpeg/clownfish.jpeg",
        package = "snic",
        mustWork = TRUE
    )
    rgb <- jpeg::readJPEG(img_path)

    # Compare grid types visually using snic_plot for immediate feedback
    types <- c("rectangular", "diamond", "hexagonal", "random")
    op <- par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
    for (tp in types) {
```

```
        seeds <- snic_grid(rgb, type = tp, spacing = 12L, padding = 18L)
        snic_plot(
            rgb,
            r = 1, g = 2, b = 3,
            seeds = seeds,
            main = paste("Grid:", tp)
        )
    }
    par(mfrow = c(1, 1))
    par(op)
}
```

`snic_grid_manual` *Interactive seed selection for SNIC segmentation*

Description

Collect seed points interactively by clicking on the image. Each left-click adds a new seed; pressing `ESC` ends the session. After each click, SNIC segmentation is recomputed and plotted for visual feedback. This is intended for exploratory and fine-tuning workflows, where automatic seeding may not be ideal.

Usage

```
snic_grid_manual(
  x,
  seeds = NULL,
  ...,
  snic_args = list(compactness = 0.5),
 snic_plot_args = list(stretch = "lin", seeds_plot_args = list(pch = 4, col = "#FFFF00",
    cex = 1), seg_plot_args = list(border = "#FFFF00", col = NA, lwd = 0.4))
)
```

Arguments

`x`	A `SpatRaster` object with a valid spatial reference and extent. Mouse clicks are interpreted in map coordinates.
`seeds`	Optional existing seed set to display and extend. If pixel coordinates are supplied, they are internally converted. If `NULL`, the seed set is initialized empty and populated interactively.
`...`	Arguments forwarded to `snic_plot` for display control. These may include `band`, `r`, `g`, `b`, `stretch`, `seeds_plot_args`, or `seg_plot_args`.
`snic_args`	A list of arguments passed to `snic`, such as `compactness`.
`snic_plot_args`	A list of display modifiers forwarded to `snic_plot` when rendering the preview.

Details

After each new seed is placed interactively, segmentation is recomputed to provide immediate feedback on how the seed placement affects clustering.

Value

A two-column data frame of seed coordinates. If `x` lacks a CRS the result is always pixel indices (`r`, `c`). When `x` has a CRS:

- If `seeds` were supplied, their coordinate system is preserved in the output.
- Otherwise the result is expressed as (`lat`, `lon`) in `EPSG:4326`.

The output can be passed directly to `snic`.

See Also

`snic`, `snic_grid`, `snic_animation`.

Examples

```
if (interactive() && requireNamespace("terra", quietly = TRUE) && terra_is_working()) {
    tiff_dir <- system.file("demo-geotiff",
        package = "snic",
        mustWork = TRUE
    )
    files <- file.path(
        tiff_dir,
        c(
            "S2_20LMR_B02_20220630.tif",
            "S2_20LMR_B04_20220630.tif",
            "S2_20LMR_B08_20220630.tif",
            "S2_20LMR_B12_20220630.tif"
        )
    )
    s2 <- terra::aggregate(terra::rast(files), fact = 8)

    seeds <- snic_grid_manual(
        s2,
        snic_args = list(compactness = 0.1),
        snic_plot_args = list(r = 4, g = 3, b = 1)
    )

    seg <- snic(s2, seeds, compactness = 0.1)

    snic_plot(
        s2,
        r = 4, g = 3, b = 1,
        stretch = "lin",
        seeds = seeds,
        seg = seg
    )
}
```

snic_plot *Plot SNIC imagery*

Description

Render image data processed by SNIC either from in-memory numeric arrays or from terra::SpatRaster objects provided by the **terra** package. The function supports plotting a single band or a three-channel RGB composite, with optional overlays for seed points and segmentation boundaries.

Usage

```
snic_plot(
  x,
  ...,
  band = 1L,
  r = NULL,
  g = NULL,
  b = NULL,
  col = getOption("snic.col", grDevices::hcl.colors(128L, "Spectral")),
  stretch = "lin",
  seeds = NULL,
 seeds_plot_args = getOption("snic.seeds_plot", list(pch = 20, col = "#00FFFF", cex =
    1)),
  seg = NULL,
 seg_plot_args = getOption("snic.seg_plot", list(border = "#FFD700", col = NA, lwd =
    0.6))
)
```

Arguments

x	Image data. For the array method this must be a numeric array with dimensions (height, width, bands). For the raster method the object must be a SpatRaster.
...	Additional arguments forwarded to the underlying plotting function. For arrays, these are passed to graphics::image(); for raster inputs they are forwarded to terra::snic_plot() (single band) or terra::plotRGB() (RGB composites).
band	Integer index of the band to display when producing a single-band plot. Defaults to the first band.
r, g, b	Integer indices (1-based) of the bands to use when composing an RGB plot. All three must be supplied to trigger RGB rendering and the image must contain at least three bands.
col	Color palette used for single-band plots. Ignored for RGB plots.
stretch	Character string indicating the contrast-stretching method. Determines how band values are scaled to the $[0, 1]$ range before plotting. One of: • "lin": linear stretch based on the minimum and maximum values (default).

- "hist": histogram equalization (redistribute values to equalize the color histogram).

Non-numeric arrays or bands with only constant values are plotted as-is.

seeds — Optional object containing seed coordinates with columns r and c. Alternately, it can have SpatRaster inputs, lat and lon columns expressed in "EPSG:4326".

seeds_plot_args — Optional named list with additional arguments passed to graphics::points() when drawing seeds. Defaults to getOption("snic.seeds_plot"), falling back to list(pch = 16, col = "#00FFFF", cex = 1) which mirrors the internal .plot_seeds() defaults.

seg — For SpatRaster inputs, an optional segmentation raster (integer labels) or already vectorized segments (a terra::SpatVector) to be drawn over the image.

seg_plot_args — Named list of arguments forwarded to terra::snic_plot() for the seg overlay. The argument add = TRUE is set automatically when not supplied. Defaults to getOption("snic.seg_plot"), falling back to list(border = "#FFD700", col = NA, lwd = 0.6) which matches the defaults used inside .plot_segments().

Value

Invisibly, NULL.

Examples

```
if (requireNamespace("terra", quietly = TRUE) && terra_is_working()) {
    tiff_dir <- system.file("demo-geotiff",
        package = "snic",
        mustWork = TRUE
    )
    files <- file.path(
        tiff_dir,
        c(
            "S2_20LMR_B02_20220630.tif",
            "S2_20LMR_B04_20220630.tif",
            "S2_20LMR_B08_20220630.tif",
            "S2_20LMR_B12_20220630.tif"
        )
    )

    # Load and optionally downsample for faster segmentation
    s2 <- terra::aggregate(terra::rast(files), fact = 8)

    # Visualize
    snic_plot(s2, r = 4, g = 3, b = 1, stretch = "lin")
}

# Simple array example using bundled JPEG
if (requireNamespace("jpeg", quietly = TRUE)) {
    img_path <- system.file("demo-jpeg/clownfish.jpeg",
        package = "snic",
        mustWork = TRUE
```

```
    )

    # Load
    rgb <- jpeg::readJPEG(img_path)

    # Visualize
    snic_plot(rgb, r = 1, g = 2, b = 3, stretch = "none")
}
```

`terra_is_working` *Check whether 'terra' CRS transformations are available*

Description

This helper attempts the same coordinate reprojection used elsewhere in the package (via `terra::project()`). When the bundled `PROJ` / `GDAL` data are missing, `terra` raises an error; here we catch it and return `FALSE` so that examples, tests, and vignettes can skip gracefully.

Usage

```
terra_is_working()
```

Details

Results are cached for the current R session to avoid repeatedly hitting `terra::project()` during a single run.

Value

A logical flag indicating whether **terra** can perform CRS transformations.

Index