

Package ‘viewscape’

June 27, 2026

Type Package

Title Viewscape Analysis

Version 3.0.0

Description Provides tools for viewscape analysis from one or multiple viewpoints using a digital surface or elevation model. Core functionality includes computing viewsheds, quantifying visual magnitude, calculating a suite of viewscape configuration metrics (extent, depth, relief, Sky View Factor, skyline variation, and patch-based landscape structure), Shannon Diversity Index and land cover feature proportions within the visible area, pairwise intervisibility networks, panoramic view generation, and visualizing results as rasters or polygons. Viewscape configuration metrics follow the methods of Tabrizian et al. (2020) <[doi:10.1016/j.landurbplan.2019.103704](https://doi.org/10.1016/j.landurbplan.2019.103704)>. The viewshed algorithm is based on Franklin & Ray (1994) <<https://api.semanticscholar.org/CorpusID:10680920>> and Wang et al. (2000) <<https://api.semanticscholar.org/CorpusID:131687018>>. Visual magnitude is derived from Chamberlain & Meitner (2013) <[doi:10.1016/j.landurbplan.2013.01.003](https://doi.org/10.1016/j.landurbplan.2013.01.003)>. Sky View Factor is computed following Oke (1981) <[doi:10.1002/joc.3370010304](https://doi.org/10.1002/joc.3370010304)> as implemented in the 'shadow' package (Dorman et al. 2019) <[doi:10.32614/RJ-2019-024](https://doi.org/10.32614/RJ-2019-024)>.

URL <https://github.com/billbillbilly/viewscape>

BugReports <https://github.com/billbillbilly/viewscape/issues>

Depends R (>= 4.2)

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), knitr, rmarkdown, greenSD, dsmSearch

Config/testthat/edition 3

VignetteBuilder knitr, rmarkdown

Imports Rcpp, rlang, methods, dplyr, sf, sp, terra, ForestTools (>= 1.0.1), parallel, pbmcapply

LinkingTo Rcpp

NeedsCompilation yes
Author Xiaohao Yang [aut, cre, cph],
Nathan Fox [aut],
Derek Van Berkel [aut],
Mark Lindquist [aut]
Maintainer Xiaohao Yang <xiaohaoy111@gmail.com>
Repository CRAN
Date/Publication 2026-06-27 09:30:02 UTC

Contents

calculate_diversity	2
calculate_feature	3
calculate_viewmetrics	4
compute_viewshed	6
fov_mask	8
intervis_network	9
pano_view	10
Viewshed-class	13
visualize_viewshed	13
visual_magnitude	14
Index	16

calculate_diversity	<i>calculate_diversity</i>
---------------------	----------------------------

Description

The calculate_diversity function is designed to calculate landscape diversity metrics within a viewshed. It takes as input a land cover raster, a viewshed object representing the observer’s line of sight, and an optional parameter to compute class proportions.

Usage

```
calculate_diversity(viewshed, land, proportion = FALSE)
```

Arguments

viewshed	Viewshed object.
land	Raster. The raster of land use/land cover representing different land use/cover classes.
proportion	logical (Optional), indicating whether to return class proportions along with the Shannon Diversity Index (SDI). (default is FALSE).

Value

List. a list containing the Shannon Diversity Index (SDI) and, if the proportion parameter is set to TRUE, a table of class proportions within the viewshed.

Examples

```
library(viewscape)
# Load a viewpoint
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# load dsm raster
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
#Compute viewshed
output <- viewscape::compute_viewshed(dsm = dsm,
                                     viewpoints = test_viewpoint,
                                     offset_viewpoint = 6, r = 1600)

# load landuse raster
test_landuse <- terra::rast(system.file("test_landuse.tif",
                                     package = "viewscape"))
diversity <- viewscape::calculate_diversity(output,
                                     test_landuse)
```

calculate_feature	<i>calculate_feature</i>
-------------------	--------------------------

Description

The calculate_feature function is designed to extract specific feature-related information within a viewshed. It allows you to compute the proportion of the feature that is present in the viewshed.

Usage

```
calculate_feature(viewshed, feature, type, exclude_value)
```

Arguments

viewshed	Viewshed object.
feature	Raster. Land cover or land use
type	Numeric. The input type of land cover raster. type=1: percentage raster (that represents the percentage of area in each cell). type=2: binary raster (that only uses two values to represent whether the feature exists in each cell).
exclude_value	Numeric. the value of those cells need to be excluded in the analysis. If type = 2, exclude_value is required.

Value

Numeric. The canopy area in the viewshed.

Examples

```
library(viewscape)
# Load a viewpoint
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# load dsm raster
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
#Compute viewshed
viewshed <- viewscape::compute_viewshed(dsm = dsm,
                                         viewpoints = test_viewpoint,
                                         offset_viewpoint = 6)

# load canopy raster
test_canopy <- terra::rast(system.file("test_canopy.tif",
                                         package = "viewscape"))

# calculate the percentage of canopy coverage
test_canopy_proportion <- viewscape::calculate_feature(viewshed = viewshed,
                                                       feature = test_canopy,
                                                       type = 2,
                                                       exclude_value = 0)
```

calculate_viewmetrics *calculate_viewmetrics*

Description

The `calculate_viewmetrics` function is designed to compute a set of configuration metrics based on a given viewshed object and optionally, digital surface models (DSM) and digital terrain models (DTM) for terrain analysis. The function calculates various metrics that describe the visibility characteristics of a landscape from a specific viewpoint. The metrics include:

1. Extent: The total area of the viewshed, calculated as the number of visible grid cells multiplied by the grid resolution
2. Depth: The furthest visible distance within the viewshed from the viewpoint
3. Vdepth: The standard deviation of distances to visible points, providing a measure of the variation in visible distances
4. Horizontal: The total visible horizontal or terrestrial area within the viewshed
5. Relief: The standard deviation of elevations of the visible ground surface
6. Sky View Factor (SVF): The proportion of visible sky hemisphere at the viewpoint, computed by casting rays in 36 azimuth directions and calculating $SVF = \text{mean}(\cos^2(\text{max obstruction angle}))$. Ranges from 0 (fully enclosed) to 1 (open sky).
7. Skyline: The standard deviation of the vertical viewscape, including visible canopy and buildings, when specified
8. Number of patches: Visible fragmentation measured by total visible patches with the viewscape
9. Mean shape index: Visible patchiness based on average perimeter-to-area ratio for all viewscape patches (vegetation and building)

10. Edge density: A measure of visible complexity based on the length of patch edges per unit area
11. Patch size: Total average size of a patches over the entire viewscape area
12. Patch density: Visible landscape granularity based on measuring patch density
13. Shannon diversity index: The abundance and evenness of land cover/use in a viewshed
14. Proportion of object: Proportion of a single type of land use or cover in a viewshed

Usage

```
calculate_viewmetrics(viewshed, dsm, dtm, masks = list())
```

Arguments

viewshed	Viewshed object.
dsm	Raster, Digital Surface Model for the calculation of
dtm	Raster, Digital Terrain Model
masks	List, a list including rasters of canopy and building footprints. For example of canopy raster, the value for cells without canopy should be 0 and the value for cells with canopy can be any number.

Value

List

References

Tabrizian, P., Baran, P.K., Berkel, D.B., Mitásová, H., & Meentemeyer, R.K. (2020). Modeling restorative potential of urban environments by coupling viewscape analysis of lidar data with experiments in immersive virtual environments. *Landscape and Urban Planning*, 195, 103704.

Dorman, M., Vulkan, A., Erell, E., & Kloog, I. (2019). shadow: Geometric Shadow Calculations. *The R Journal*, 11(1), 287–309. <https://github.com/michaeldorman/shadow>

Oke, T. R. (1981). Canyon geometry and the nocturnal urban heat island: comparison of scale model and field observations. *Journal of Climatology*, 1(3), 237–254.

Examples

```
# Load in DSM
test_dsm <- terra::rast(system.file("test_dsm.tif",
                                     package = "viewscape"))

# Load DTM
test_dtm <- terra::rast(system.file("test_dtm.tif",
                                     package = "viewscape"))

# Load canopy raster
test_canopy <- terra::rast(system.file("test_canopy.tif",
                                       package = "viewscape"))

# Load building footprints raster
```

```

test_building <- terra::rast(system.file("test_building.tif",
                                         package = "viewscape"))

# Load in the viewpoint
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp",
                                         package = "viewscape"))

# Compute viewshed
output <- viewscape::compute_viewshed(dsm = test_dsm,
                                      viewpoints = test_viewpoint,
                                      offset_viewpoint = 6, r = 1600)

# calculate metrics given the viewshed, canopy, and building footprints
test_metrics <- viewscape::calculate_viewmetrics(output,
                                                  test_dsm,
                                                  test_dtm,
                                                  list(test_canopy, test_building))

```

compute_viewshed	<i>compute_viewshed</i>
------------------	-------------------------

Description

The `compute_viewshed` function is designed for computing viewsheds, which are areas visible from specific viewpoints, based on a Digital Surface Model (DSM). It provides flexibility for single or multi-viewpoint analyses and allows options for parallel processing, raster output, and plotting.

Usage

```

compute_viewshed(
  dsm,
  viewpoints,
  offset_viewpoint = 1.7,
  offset_height = 0,
  r = NULL,
  refraction_factor = 0.13,
  method = "plane",
  parallel = FALSE,
  workers = 1,
  raster = FALSE,
  plot = FALSE
)

```

Arguments

dsm	Raster, the digital surface model/digital elevation model
-----	---


```

                                offset_viewpoint = 6)
# subset viewshed using the field of view (north-facing 120° arc)
out <- viewscape::fov_mask(viewshed, c(-60, 60))

```

intervis_network	<i>intervis_network</i>
------------------	-------------------------

Description

Computes an intervisibility network among a set of viewpoints, determining which pairs of viewpoints have a direct line of sight to each other. Results can be returned as a binary adjacency matrix or as an sf collection of lines suitable for mapping the network.

Usage

```

intervis_network(
  viewpoints,
  dsm,
  offset_viewpoint = 1.7,
  r = NULL,
  refraction_factor = 0.13,
  method = "plane",
  output = "matrix"
)

```

Arguments

viewpoints	sf point(s), or a matrix/data.frame of x,y coordinates. At least 2 viewpoints are required.
dsm	SpatRaster. A digital surface / elevation model.
offset_viewpoint	Numeric. Observer height above the surface in metres . Either a single value applied to all viewpoints, or a vector of per-viewpoint heights. Default is 1.7 m. Automatically converted to the CRS unit of the DSM.
r	Numeric (optional). Analysis radius in metres . Pairs separated by more than r are treated as non-visible. Default is 1000 m. Automatically converted to the CRS unit of the DSM.
refraction_factor	Numeric. Atmospheric refraction coefficient. Default is 0.13.
method	Character. Viewshed algorithm: "plane" (default) or "los". See compute_viewshed for details.
output	Character. Return type: "matrix" (default) returns an N x N integer adjacency matrix where row i, column j is 1 if viewpoint i can see viewpoint j, and 0 otherwise (NA on the diagonal); "lines" returns an sf data frame of LINESTRINGs connecting visible pairs, with columns from, to, and mutual (TRUE when both directions are visible).

Value

An N x N integer matrix (output = "matrix") or an sf data frame of lines (output = "lines").

See Also

`compute_viewshed()` `fov_mask()`

Examples

```
library(viewscape)
# Load viewpoints
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# Load DSM
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
# Compute intervisibility matrix
mat <- intervis_network(test_viewpoint, dsm, offset_viewpoint = 1.7)
# Compute intervisibility as lines
net <- intervis_network(test_viewpoint, dsm, offset_viewpoint = 1.7,
                        output = "lines")
```

pano_view

pano_view

Description

Generate an equirectangular or cylindrical panoramic view from a DSM.

For method = "equirectangular":

- **No semantic:** automatically downloads ESRI WorldImagery tiles via **greenSD** and projects the satellite RGB colours onto the panoramic rays, producing a 3-band colour panorama (R, G, B, 0–255).
- **With semantic:** projects the supplied land-use / land-cover raster onto the panoramic rays, returning a 2-layer SpatRaster (depth + land-cover codes).

For method = "cylindrical" the function always returns a single-layer distance panorama (with an optional second semantic layer).

Usage

```
pano_view(
  dsm = NULL,
  vpt = NULL,
  h = 6,
  semantic = NULL,
  method = "equirectangular",
  sky_threshold = 3,
```

```

    step_size = 0.5,
    max_dist = 500,
    pano_dim = c(256, 512),
    ares = 0.0174533,
    heading = 0,
    color_source = NULL,
    sky_color = c(135, 206, 235),
    satellite_zoom = 17L,
    plot = FALSE,
    legend = TRUE,
    axes = TRUE
)

```

Arguments

dsm	A SpatRaster object (single layer) of surface elevation.
vpt	A POINT sf object or numeric coordinate pair representing the viewpoint (in projected coordinates).
h	Numeric. Height of the viewpoint above the ground surface (in metres). Default: 6.
semantic	A SpatRaster object (single layer) of land-use / land-cover class values. When supplied with method = "equirectangular", the panorama encodes the land-cover code at each ray-hit cell (depth layer plus semantic layer). Binary 0/1 rasters are treated as occupied vertical structures. When supplied with method = "cylindrical", the return value has two layers: depth and semantic class.
method	Character. "equirectangular" (default) or "cylindrical".
sky_threshold	Numeric. Elevation buffer (metres) below the ray position that is still treated as terrain (default: 3.0). Must be less than h; if not, it is automatically clamped to $h - 0.01$.
step_size	Numeric. Ray marching step size in DSM coordinate units (default: 0.5).
max_dist	Numeric. Maximum ray distance (default: 500).
pano_dim	Integer vector of length 2: panorama height and width in pixels (default: c(256, 512)).
ares	Numeric. Cylindrical vertical angular resolution in radians per pixel. Ignored when method = "equirectangular" (default: 0.0174533).
heading	Numeric. Camera heading in degrees (default: 0 = north). See Details.
color_source	Optional SpatRaster with exactly 3 bands (R, G, B, values 0–255). Overrides the automatic greenSD satellite download when method = "equirectangular" and semantic = NULL. Useful when you already have a local satellite image.
sky_color	Numeric vector of length 3 (R, G, B, 0–255). Colour painted on sky pixels in the colour panorama. Default: sky blue c(135, 206, 235).
satellite_zoom	Integer. Tile zoom level passed to greenSD::get_tile_green() when auto-fetching satellite imagery (default: 17, approximately 1.2 m/px at 42 degrees N).
plot	Logical. Whether to plot the result (default: FALSE).

legend	Logical. Whether to display a legend when plotting (default: TRUE).
axes	Logical. Whether to display axes when plotting (default: TRUE).

Details

For heading:

- heading = 0 → facing north (default)
- heading = 90 → facing east
- heading = 180 → facing south
- heading = 270 → facing west

The equirectangular colour panorama requires the **greenSD** package (install with `devtools::install_github("billbillb/greenSD")`). It calls `greenSD::get_tile_green(bbox, zoom, provider = "esri")` and uses the raw RGB tile (`[[2]]`) as the colour source.

Value

- method = "equirectangular", no semantic: 3-band SpatRaster (R, G, B, 0–255). Plot with `terra::plotRGB()`.
- method = "equirectangular", with semantic: 2-layer SpatRaster (depth in metres, land-cover code).
- method = "cylindrical", no semantic: single-layer distance SpatRaster (NA = sky).
- method = "cylindrical", with semantic: 2-layer SpatRaster (depth, semantic class).

References

Lu, X., Li, Z., Cui, Z., Oswald, M. R., Pollefeys, M., & Qin, R. (2020). Geometry-aware satellite-to-ground image synthesis for urban areas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 859-867).

Examples

```
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
vpt <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# cylindrical distance panorama (no internet needed)
result <- viewscape::pano_view(dsm, vpt, h = 6, method = "cylindrical")
```

Viewshed-class	<i>An S4 class to represent the viewshed</i>
----------------	--

Description

A viewshed object contains a 'matrix' of visible and invisible area, resolution, extent, and crs

Slots

visible matrix
 resolution vector
 extent numeric
 crs character

visualize_viewshed	<i>visualize_viewshed</i>
--------------------	---------------------------

Description

The visualize_viewshed function is designed for the visualization of a viewshed analysis, providing users with various options for visualizing the results. The function works with a viewshed object and offers multiple plotting and output types.

Usage

```
visualize_viewshed(viewshed, plottype = "", outputtype = "")
```

Arguments

viewshed	Viewshed object
plottype	Character, specifying the type of visualization ("polygon" or "raster").
outputtype	Character, specifying the type of output object ("raster" or "polygon").

Value

Visualized viewshed as either a raster or polygon object, depending on the outputtype specified.

Examples

```
# Load a viewpoint
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# load dsm raster
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
#Compute viewshed
viewshed <- compute_viewshed(dsm = dsm,
                             viewpoints = test_viewpoint,
                             offset_viewpoint = 6)
# Visualize the viewshed as polygons
visualize_viewshed(viewshed, plottype = "polygon")
# Visualize the viewshed as a raster
visualize_viewshed(viewshed, plottype = "raster")
# Get the visualized viewshed as a polygon object
polygon_viewshed <- visualize_viewshed(viewshed,
                                       plottype = "polygon",
                                       outputtype = "polygon")
```

visual_magnitude

visual_magnitude

Description

Visual Magnitude quantifies the extent of a visible region as perceived by an observer. It is derived from the surface's slope and angle features, alongside the observer's relative distance from the area (Chamberlain & Meitner).

Usage

```
visual_magnitude(viewshed, dsm)
```

Arguments

viewshed	Viewshed object.
dsm	SpatRaster. A digital surface / elevation model

Value

SpatRaster

References

Chamberlain, B. C., & Meitner, M. J. (2013). A route-based visibility analysis for landscape management. *Landscape and Urban Planning*, 111, 13-24.

See Also

[compute_viewshed\(\)](#)

Examples

```
# Load a viewpoint
test_viewpoint <- sf::read_sf(system.file("test_viewpoint.shp", package = "viewscape"))
# load dsm raster
dsm <- terra::rast(system.file("test_dsm.tif", package = "viewscape"))
# Compute viewshed
viewshed <- viewscape::compute_viewshed(dsm = dsm,
                                         viewpoints = test_viewpoint,
                                         offset_viewpoint = 6)

# Compute visual magnitude
vm <- viewscape::visual_magnitude(viewshed, dsm)
```

Index

`calculate_diversity`, [2](#)
`calculate_feature`, [3](#)
`calculate_viewmetrics`, [4](#)
`compute_viewshed`, [6](#), [9](#)
`compute_viewshed()`, [8](#), [10](#), [14](#)

`fov_mask`, [8](#)
`fov_mask()`, [10](#)

`intervis_network`, [9](#)

`pano_view`, [10](#)

`Viewshed-class`, [13](#)
`visual_magnitude`, [14](#)
`visualize_viewshed`, [13](#)