# Package 'viewscape'

July 22, 2025

**Type** Package

**Title** Viewscape Analysis

**Version** 2.0.2

**Description** A collection of functions to make R a more effective viewscape analysis tool for calculating viewscape metrics based on computing the viewable area for given a point/multiple viewpoints and a digital elevation model.The method of calculating viewscape metrics implemented in this package are based on the work of Tabrizian et al. (2020) <doi:10.1016/j.landurbplan.2019.103704>. The algorithm of computing viewshed is based on the work of Franklin & Ray. (1994) <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=555780f6f5d7e537eb1edb28862c86d1519af2be>.

**URL** https://github.com/land-info-lab/viewscape

**BugReports** https://github.com/land-info-lab/viewscape/issues

**Depends** R (>= 4.2)

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Language** en-US

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

**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 <xiaohaoy@umich.edu>

**Repository** CRAN

**Date/Publication** 2024-12-19 21:20:02 UTC

# Contents



---

`calculate_diversity` *calculate_diversity*

---

## 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` *calculate_feature*

---

## 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 reqired. |

## 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. Skyline: The standard deviation of the vertical viewscape, including visible canopy and buildings, when specified
7. Number of patches: Visible fragmentation measured by total visible patches with the viewscape
8. Mean shape index: Visible patchiness based on average perimeter-to-area ratio for all viewscape patches (vegetation and building)
9. Edge density: A measure of visible complexity based on the length of patch edges per unit area
10. Patch size: Total average size of a patches over the entire viewscape area

11. Patch density: Visible landscape granularity based on measuring patch density
12. Shannon diversity index: The abundance and evenness of land cover/use in a viewshed
13. 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.

## 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` *compute_viewshed*

---

## 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 |
| `viewpoints` | sf point(s) or vector including x,y coordinates of a viewpoint or a matrix including several viewpoints with x,y coordinates |
| `offset_viewpoint` | numeric, setting the height of the viewpoint. (default is 1.7 meters). |
| `offset_height` | numeric, setting the height of positions that a given viewpoint will look at. (defaut is 0) |
| `r` | Numeric (optional), setting the radius for viewshed analysis. (The default is 1000m/3281ft) |

`refraction_factor`
: Number, indicating the refraction factor. The refraction factor adjusts the effect of atmospheric refraction on the apparent curvature of the Earth. In most standard applications, a refraction factor of 0.13 is used, and so does this function. However, the appropriate refraction factor may vary depending on environmental conditions.

`method`
: Character, The algorithm for computing a viewshed: "plane" and "los" (see details). "plane" is used as default.

`parallel`
: Logical, (default is FALSE) indicating if parallel computing should be used to compute viewsheds of multiview points. When it is TRUE, arguements 'raster' and 'plot' are ignored

`workers`
: Numeric, indicating the number of CPU cores that will be used for parallel computing. It is required if 'parallel' is 'TRUE'.

`raster`
: Logical, (default is FALSE) if it is TRUE, the raster of viewshed will be returned. The default is FALSE

`plot`
: Logical, (default is FALSE) if it is TRUE, the raster of viewshed will be displayed

## Details

For method, "plane" is the reference plane algorithm introduced by Wang et al. (2000) and "los" is the line of sight algorithm (Franklin & Ray, 1994). The reference plane algorithm can be more time-efficient than the line of sight algorithm, whereas the accuracy of the line of sight is better.

## Value

Raster or list. For single-viewpoint analysis, the function returns either a raster (raster is TRUE) or a viewshed object. Value 1 means visible while value 0 means invisible. For multi-viewpoint analysis, a list of viewsheds is returned.

## References

Franklin, W. R., & Ray, C. (1994, May). Higher isn't necessarily better: Visibility algorithms and experiments. In Advances in GIS research: sixth international symposium on spatial data handling (Vol. 2, pp. 751-770). Edinburgh: Taylor & Francis.

Wang, J., Robinson, G. J., & White, K. (2000). Generating viewsheds without using sightlines. Photogrammetric engineering and remote sensing, 66(1), 87-90.

## See Also

`fov_mask()` `visual_magnitude()`

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

---

`fov_mask` *fov_mask*

---

## Description

The fov_mask function is designed to subset a viewshed based on its viewpoint and the field of view

## Usage

```
fov_mask(viewshed, fov)
```

## Arguments

| | |
|---|---|
| `viewshed` | viewshed object, generated by compute_viewshed() |
| `fov` | Vector, specifying the field of view with two angles in degree (e.g. c(10,100)) for masking a viewshed based on its viewpoints. See details. |

## Details

For defining the field of view ('fov'), angles range from 0 to 360 degrees, with 0 inclusive and 360 exclusive. The initial angle must be smaller than the terminal angle in the sequence c(a,b) (a < b). To capture the northeast quadrant of a viewshed, one would use c(0,90), while the eastern quadrant would be delineated by c(45,315) as shown below:

| | | |
|---|---|---|
| 135 | 90 | 45 |
| 180 | v | 0 |
| 225 | 270 | 315 |

Here, 'v' represents the viewpoint, with angles measured counterclockwise from due north.

## Value

viewshed object

## 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,
                                        viewpoints = test_viewpoint,
                                        offset_viewpoint = 6)
# subset viewshed using the field of view
out <- viewscape::fov_mask(viewshed, c(40,160))
```

---

# Viewshed-class *An S4 class to represent the viewshed*

---

## 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 *visualize_viewshed*

---

## 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

This function is still in progress. 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` | Raster, the 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