Package: slopes (via r-universe)

July 17, 2024

Title Calculate Slopes of Roads, Rivers and Trajectories

Version 1.0.1

Description Functions and example data to support research into the slope (also known as longitudinal gradient or steepness) of linear geographic entities such as roads <doi:10.1038/s41597-019-0147-x> and rivers <doi:10.1016/j.jhydrol.2018.06.066>. The package was initially developed to calculate the steepness of street segments but can be used to calculate steepness of any linear feature that can be represented as LINESTRING geometries in the 'sf' class system. The package takes two main types of input data for slope calculation: vector geographic objects representing linear features, and raster geographic objects with elevation values (which can be downloaded using functionality in the package) representing a continuous terrain surface. Where no raster object is provided the package attempts to download elevation data using the 'ceramic' package.

License GPL-3

URL https://github.com/ropensci/slopes/, https://docs.ropensci.org/slopes/

BugReports https://github.com/ropensci/slopes/issues

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports sf, raster, methods, pbapply, geodist, colorspace

Depends R (>= 2.10)

Suggests terra, knitr, rmarkdown, ceramic, bookdown, covr, testthat, osmextract, stplanr, dplyr, rgdal, tmap, leaflet, bench

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/ropensci/slopes

RemoteRef master

RemoteSha c45cf51b5ff44821bc188c6fdb55f04c7c82cd26

Contents

cyclestreets_route 2
dem_lisbon_raster 3
elevation_add 4
elevation_extract 5
elevation_get 6
lisbon_road_network 7
lisbon_road_segment 8
lisbon_route 9
magnolia_xy 9
plot_dz 10
plot_slope 11
sequential_dist 13
slope_matrix 13
slope_raster 15
slope_vector 16
slope_xyz 17
z_value 18

Index **20**

`cyclestreets_route` *A journey from CycleStreets.net*

Description

Road segments representing suggested route to cycle in Leeds, UK.

Usage

```
cyclestreets_route
```

Format

An object of class `sf` with 18 rows and 14 columns on route characteristics. See https://rpackage.cyclestreets.net/reference/jou
for details.

Details

Simple feature collection with 30 features and 32 fields

See `data-raw/cyclestreets_route.R` in the package's github repo for details.

Source

CycleStreets.net

Examples

```
library(sf)
class(cyclestreets_route)
plot(cyclestreets_route$geometry)
cyclestreets_route
```

---

dem_lisbon_raster *Elevation in central Lisbon, Portugal*

Description

A dataset containing elevation in and around Lisbon with a geographic resolution of 10m. The dataset is 200 pixels wide by 133 pixels high, covering 2.7 square kilometres of central Lisbon.

Usage

```
dem_lisbon_raster
```

Format

A raster dataset containing elevation above sea level in a 1km bounding box in Lisbon, Portugal.

Details

The dataset was acquired by Instituto Superior Técnico (University of Lisbon) in 2012, covers all the Northern Metropolitan Area of Lisbon, and has a 10m cell resolution, when projected at the official Portuguese EPSG: 3763 - TM06/ETRS89. The dataset was released as an open access dataset with permission from the University of Lisbon to support this project.

Source

https://github.com/rspatial/terra/issues/29

Examples

```
library(sf)
library(raster)
dim(dem_lisbon_raster)
res(dem_lisbon_raster)
names(dem_lisbon_raster)
plot(dem_lisbon_raster)
plot(lisbon_road_network["Avg_Slope"], add = TRUE)
```

elevation_add *Take a linestring and add a third (z) dimension to its coordinates*

Description

Take a linestring and add a third (z) dimension to its coordinates

Usage

```
elevation_add(
  routes,
  dem = NULL,
  method = "bilinear",
  terra = has_terra() && methods::is(dem, "SpatRaster")
)
```

Arguments

routes	Routes, the gradients of which are to be calculated. The object must be of class `sf` or `sfc` with `LINESTRING` geometries.
dem	Raster overlapping with `routes` and values representing elevations
method	The method of estimating elevation at points, passed to the `extract` function for extracting values from raster datasets. Default: `"bilinear"`.
terra	Should the `terra` package be used? `TRUE` by default if the package is installed *and* if `dem` is of class `SpatRast`

Value

An sf object that is identical to the input `routes`, except that the coordinate values in the ouput has a third `z` dimension representing the elevation of each vertex that defines a linear feature such as a road.

Examples

```
library(sf)
routes = lisbon_road_network[204, ]
dem = dem_lisbon_raster
(r3d = elevation_add(routes, dem))
library(sf)
st_z_range(routes)
st_z_range(r3d)
plot(st_coordinates(r3d)[, 3])
plot_slope(r3d)

# Get elevation data (requires internet connection and API key):
r3d_get = elevation_add(cyclestreets_route)
plot_slope(r3d_get)
```

`elevation_extract` *Extract elevations from coordinates*

Description

This function takes a series of points located in geographical space and a digital elevation model as inputs and returns a vector of elevation estimates associated with each point. The function takes locations represented as a matrix of XY (or longitude latitude) coordinates and a digital elevation model (DEM) with class `raster` or `terra`. It returns a vector of values representing estimates of elevation associated with each of the points.

Usage

```
elevation_extract(
  m,
  dem,
  method = "bilinear",
  terra = has_terra() && methods::is(dem, "SpatRaster")
)
```

Arguments

`m`	Matrix containing coordinates and elevations or an sf object representing a linear feature.
`dem`	Raster overlapping with `routes` and values representing elevations
`method`	The method of estimating elevation at points, passed to the `extract` function for extracting values from raster datasets. Default: `"bilinear"`.
`terra`	Should the `terra` package be used? `TRUE` by default if the package is installed *and* if `dem` is of class `SpatRast`

Details

By default, the elevations are estimated using bilinear interpolation (`method = "bilinear"`) which calculates point height based on proximity to the centroids of surrounding cells. The value of the `method` argument is passed to the `method` argument in `raster::extract()` or `terra::extract()` depending on the class of the input raster dataset.

See Kidner et al. (1999) for descriptions of alternative elevation interpolation and extrapolation algorithms.

Value

A vector of elevation values.

References

Kidner, David, Mark Dorey, and Derek Smith. "What's the point? Interpolation and extrapolation with a regular grid DEM." Fourth International Conference on GeoComputation, Fredericksburg, VA, USA. 1999.

Examples

```
dem = dem_lisbon_raster
elevation_extract(lisbon_road_network[1, ], dem)
m = sf::st_coordinates(lisbon_road_network[1, ])
elevation_extract(m, dem)
elevation_extract(m, dem, method = "simple")
# Test with terra (requires internet connection):

if(slopes:::has_terra()) {
et = terra::rast(dem_lisbon_raster)
elevation_extract(m, et)
}
```

---

elevation_get — *Get elevation data from hosted maptile services*

Description

`elevation_get()` uses the `cc_elevation()` function from the `ceramic` package to get DEM data in raster format anywhere worldwide. It requires an API that can be added by following guidance in the package's README and in the `slopes` vignette.

Usage

```
elevation_get(routes, ..., output_format = "raster")
```

Arguments

`routes`	Routes, the gradients of which are to be calculated. The object must be of class `sf` or `sfc` with `LINESTRING` geometries.
`...`	Options passed to `cc_elevation()`
`output_format`	What format to return the data in? Accepts `"raster"` (the default) and `"terra"`.

Details

Note: if you use the `cc_elevation()` function directly to get DEM data, you can cache the data, as described in the package's README.

Value

A raster object with cell values representing elevations in the bounding box of the input `routes` object.

Examples

```
# Time-consuming examples that require an internet connection and API key:

library(sf)
library(raster)
routes = cyclestreets_route
e = elevation_get(routes)
class(e)
crs(e)
e
plot(e)
plot(st_geometry(routes), add = TRUE)
```

`lisbon_road_network` *Road segments in Lisbon*

Description

A dataset representing road segments in Lisbon, with X, Y and Z (elevation) dimensions for each coordinate.

Usage

`lisbon_road_network`

Format

An object of class `sf`, key variables of which include

OBJECTID ID of the object

Z_Min The minimum elevation on the linear feature from ArcMAP

Z_Max The max elevation on the linear feature from ArcMAP

Z_Mean The mean elevation on the linear feature from ArcMAP

Slope_Min The minimum slope on the linear feature from ArcMAP

Slope_Max The max slope on the linear feature from ArcMAP

Slope_Mean The mean slope on the linear feature from ArcMAP

geom The geometry defining the LINESTRING component of the segment

Details

The dataset covers 32 km of roads in central Lisbon, overlapping with the area covered by the `dem_lisbon_raster` dataset.

Source

Produced by ESRI's 3D Analyst extension

Examples

```
library(sf)
names(lisbon_road_network)
sum(st_length(lisbon_road_network))
plot(lisbon_road_network["Avg_Slope"])
```

---

lisbon_road_segment *A road segment in Lisbon, Portugal*

Description

A single road segment and a 3d version. Different versions of this dataset are provided.

Usage

```
lisbon_road_segment
```

Format

An object of class `sf`

Details

The `lisbon_road_segment` has 23 columns and 1 row.

The `lisbon_road_segment_xyz_mapbox` was created with: `lisbon_road_segment_xyz_mapbox = elevation_add(lisbon_road_segment)`.

Source

Produced by ESRI's 3D Analyst extension

Examples

```
lisbon_road_segment
lisbon_road_segment_3d
lisbon_road_segment_xyz_mapbox
```

`lisbon_route` *A route composed of a single linestring in Lisbon, Portugal*

Description

A route representing a trip from the Santa Catarina area in the East of central Lisbon the map to the Castelo de São Jorge in the West of central Lisbon.

Usage

```
lisbon_route
```

Format

An object of class `sf`

Details

Different versions of this dataset are provided.

The `lisbon_route` object has 1 row and 4 columns: geometry, ID, length and whether or not a path was found.

The `lisbon_route_xyz_mapbox` was created with: `lisbon_route_xyz_mapbox = elevation_add(lisbon_route)`.

Source

See the `lisbon_route.R` script in `data-raw`

Examples

```
lisbon_route
lisbon_route_3d
lisbon_route_xyz_mapbox
```

`magnolia_xy` *Road segments in Magnolia, Seattle*

Description

A dataset representing road segments in the Magnolia area of Seattle with X, Y and Z (elevation) dimensions for each coordinate.

Usage

```
magnolia_xy
```

Format

An object of class sf

Source

Accessed in early 2021 from the seattle-streets layer from the data-seattlecitygis website.

Examples

```
names(magnolia_xy)
plot(magnolia_xy["SLOPE_PCT"])
```

---

plot_dz *Plot a digital elevation profile based on xyz data*

Description

Plot a digital elevation profile based on xyz data

Usage

```
plot_dz(
  d,
  z,
  fill = TRUE,
  horiz = FALSE,
  pal = colorspace::diverging_hcl,
  ...,
  legend_position = "top",
  col = "black",
  cex = 0.9,
  bg = grDevices::rgb(1, 1, 1, 0.8),
  title = "Slope colors (percentage gradient)",
  brks = NULL,
  seq_brks = NULL,
  ncol = 4
)
```

Arguments

d	Cumulative distance
z	Elevations at points across a linestring
fill	Should the profile be filled? TRUE by default
horiz	Should the legend be horizontal (FALSE by default)
pal	Color palette to use, colorspace::diverging_hcl by default.
...	Additional parameters to pass to legend

legend_position
: The legend position. One of "bottomright", "bottom", "bottomleft", "left", "topleft", "top" (the default), "topright", "right" and "center".

col
: Line colour, black by default

cex
: Legend size, 0.9 by default

bg
: Legend background colour, `grDevices::rgb(1, 1, 1, 0.8)` by default.

title
: Title of the legend, `NULL` by default.

brks
: Breaks in colour palette to show. `c(1, 3, 6, 10, 20, 40, 100)` by default.

seq_brks
: Sequence of breaks to show in legend. Includes negative numbers and omits zero by default

ncol
: Number of columns in legend, 4 by default.

Value

A plot showing the elevation profile associated with a linestring.

Examples

```
library(sf)
route_xyz = lisbon_road_segment_3d
m = st_coordinates(route_xyz)
d = cumsum(sequential_dist(m, lonlat = FALSE))
d = c(0, d)
z = m[, 3]
slopes:::plot_dz(d, z, brks = c(3, 6, 10, 20, 40, 100))
```

---

plot_slope *Plot slope data for a 3d linestring with base R graphics*

Description

Plot slope data for a 3d linestring with base R graphics

Usage

```
plot_slope(
  route_xyz,
  lonlat = sf::st_is_longlat(route_xyz),
  fill = TRUE,
  horiz = FALSE,
  pal = colorspace::diverging_hcl,
  legend_position = "top",
  col = "black",
  cex = 0.9,
  bg = grDevices::rgb(1, 1, 1, 0.8),
  title = "Slope colors (percentage gradient)",
```

```
    brks = c(3, 6, 10, 20, 40, 100),
    seq_brks = seq(from = 3, to = length(brks) * 2 - 2),
    ncol = 4,
    ...
  )
```

Arguments

`route_xyz`	An sf linestring with x, y and z coordinates, representing a route or other linear object.
`lonlat`	Are the routes provided in longitude/latitude coordinates? By default, value is from the CRS of the routes (`sf::st_is_longlat(routes)`).
`fill`	Should the profile be filled? `TRUE` by default
`horiz`	Should the legend be horizontal (`FALSE` by default)
`pal`	Color palette to use, `colorspace::diverging_hcl` by default.
`legend_position`	The legend position. One of "bottomright", "bottom", "bottomleft", "left", "topleft", "top" (the default), "topright", "right" and "center".
`col`	Line colour, black by default
`cex`	Legend size, 0.9 by default
`bg`	Legend background colour, `grDevices::rgb(1, 1, 1, 0.8)` by default.
`title`	Title of the legend, `NULL` by default.
`brks`	Breaks in colour palette to show. `c(1, 3, 6, 10, 20, 40, 100)` by default.
`seq_brks`	Sequence of breaks to show in legend. Includes negative numbers and omits zero by default
`ncol`	Number of columns in legend, 4 by default.
`...`	Additional parameters to pass to legend

Value

A plot showing the elevation profile associated with a linestring.

Examples

```
plot_slope(lisbon_route_3d)
route_xyz = lisbon_road_segment_3d
plot_slope(route_xyz)
plot_slope(route_xyz, brks = c(1, 2, 4, 8, 16, 30))
plot_slope(route_xyz, s = 5:8)
```

sequential_dist — *Calculate the sequential distances between sequential coordinate pairs*

Description

Set `lonlat` to `FALSE` if you have projected data, e.g. with coordinates representing distance in meters, not degrees. Lonlat coodinates are assumed (`lonlat = TRUE` is the default).

Usage

```
sequential_dist(m, lonlat = TRUE)
```

Arguments

`m`	Matrix containing coordinates and elevations. The matrix should have three columns: x, y, and z, in that order. Typically these correspond to location in the West-East, South-North, and vertical elevation axes respectively. In data with geographic coordinates, Z values are assumed to be in metres. In data with projected coordinates, Z values are assumed to have the same units as the X and Y coordinates.
`lonlat`	Are the coordinates in lon/lat (geographic) coordinates? TRUE by default.

Value

A vector of distance values in meters if `lonlat = TRUE` or the map units of the input data if `lonlat = FALSE` between consecutive vertices.

Examples

```
x = c(0, 2, 3, 4, 5, 9)
y = c(0, 0, 0, 0, 0, 1)
m = cbind(x, y)
d = sequential_dist(m, lonlat = FALSE)
d
nrow(m)
length(d)
```

slope_matrix — *Calculate the gradient of line segments from a 3D matrix of coordinates*

Description

Calculate the gradient of line segments from a 3D matrix of coordinates

Usage

```
slope_matrix(m, elevations = m[, 3], lonlat = TRUE)

slope_matrix_mean(m, elevations = m[, 3], lonlat = TRUE, directed = FALSE)

slope_matrix_weighted(m, elevations = m[, 3], lonlat = TRUE, directed = FALSE)
```

Arguments

m	Matrix containing coordinates and elevations. The matrix should have three columns: x, y, and z, in that order. Typically these correspond to location in the West-East, South-North, and vertical elevation axes respectively. In data with geographic coordinates, Z values are assumed to be in metres. In data with projected coordinates, Z values are assumed to have the same units as the X and Y coordinates.
elevations	Elevations in same units as x (assumed to be metres). Default value: `m[, 3]`, meaning the 'z' coordinate in a matrix of coordinates.
lonlat	Are the coordinates in lon/lat (geographic) coordinates? TRUE by default.
directed	Should the value be directed? `FALSE` by default. If `TRUE` the result will be negative when it represents a downslope (when the end point is lower than the start point).

Value

A vector of slope gradients associated with each linear element (each line between consecutive vertices) associated with linear features. Returned values for `slope_matrix_mean()` and `slope_matrix_weighted()` are summary statistics for all linear elements in the linestring. The output value is a proportion representing the change in elevation for a given change in horizontal movement along the linestring. 0.02, for example, represents a low gradient of 2% while 0.08 represents a steep gradient of 8%.

Examples

```
x = c(0, 2, 3, 4, 5, 9)
y = c(0, 0, 0, 0, 0, 9)
z = c(1, 2, 2, 4, 3, 0) / 10
m = cbind(x, y, z)
slope_matrix_weighted(m, lonlat = FALSE)
slope_matrix_weighted(m, lonlat = FALSE, directed = TRUE)
# 0 value returned if no change in elevation:
slope_matrix_weighted(m,lonlat = FALSE, directed = TRUE,
  elevations = c(1, 2, 2, 4, 3, 1))
slope_matrix_mean(m, lonlat = FALSE)
slope_matrix_mean(m, lonlat = FALSE, directed = TRUE)
plot(x, z, ylim = c(-0.5, 0.5), type = "l")
(gx = slope_vector(x, z))
(gxy = slope_matrix(m, lonlat = FALSE))
abline(h = 0, lty = 2)
points(x[-length(x)], gx, col = "red")
points(x[-length(x)], gxy, col = "blue")
```

```
title("Distance (in x coordinates) elevation profile",
  sub = "Points show calculated gradients of subsequent lines")
```

---

slope_raster *Calculate the gradient of line segments from a raster dataset*

Description

This function takes an `sf` representing routes over geographical space and a raster dataset representing the terrain as inputs. It returns the average gradient of each route feature.

Usage

```
slope_raster(
  routes,
  dem,
  lonlat = sf::st_is_longlat(routes),
  method = "bilinear",
  fun = slope_matrix_weighted,
  terra = has_terra() && methods::is(dem, "SpatRaster"),
  directed = FALSE
)
```

Arguments

`routes`	Routes, the gradients of which are to be calculated. The object must be of class `sf` or `sfc` with `LINESTRING` geometries.
`dem`	Raster overlapping with `routes` and values representing elevations
`lonlat`	Are the routes provided in longitude/latitude coordinates? By default, value is from the CRS of the routes (`sf::st_is_longlat(routes)`).
`method`	The method of estimating elevation at points, passed to the `extract` function for extracting values from raster datasets. Default: `"bilinear"`.
`fun`	The slope function to calculate per route, `slope_matrix_weighted` by default.
`terra`	Should the `terra` package be used? `TRUE` by default if the package is installed *and* if `dem` is of class `SpatRast`
`directed`	Should the value be directed? `FALSE` by default. If `TRUE` the result will be negative when it represents a downslope (when the end point is lower than the start point).

Details

If calculating slopes associated with OSM data, the results may be better if the network is first split-up, e.g. using the function `stplanr::rnet_breakup_vertices()` from the stplanr package. **Note:** The `routes` object must have a geometry type of `LINESTRING`. The `sf::st_cast()` function can convert from `MULTILINESTRING` (and other) geometries to `LINESTRING`s as follows: `r_linestring = sf::st_cast(routes, "LINESTRING")`.

Value

A vector of slopes equal in length to the number simple features (rows representing linestrings) in the input object.

Examples

```
library(sf)
routes = lisbon_road_network[1:3, ]
dem = dem_lisbon_raster
(s = slope_raster(routes, dem))
cor(routes$Avg_Slope, s)
slope_raster(routes, dem, directed = TRUE)
# Demonstrate that reverse routes have the opposite directed slope
slope_raster(st_reverse(routes), dem, directed = TRUE)
```

---

slope_vector — *Calculate the gradient of line segments from distance and elevation vectors*

Description

`slope_vector()` calculates the slopes associated with consecutive elements in one dimensional distance and associated elevations (see examples).

`slope_distance()` calculates the slopes associated with consecutive distances and elevations.

`slope_distance_mean()` calculates the mean average slopes associated with consecutive distances and elevations.

`slope_distance_weighted()` calculates the slopes associated with consecutive distances and elevations, with the mean value associated with each set of distance/elevation vectors weighted in proportion to the distance between each elevation measurement, so longer sections have proportionally more influence on the resulting gradient estimate (see examples).

Usage

```
slope_vector(x, elevations)

slope_distance(d, elevations)

slope_distance_mean(d, elevations, directed = FALSE)

slope_distance_weighted(d, elevations, directed = FALSE)
```

Arguments

x	Vector of locations
elevations	Elevations in same units as x (assumed to be metres)
d	Vector of distances between points

directed — Should the value be directed? `FALSE` by default. If `TRUE` the result will be negative when it represents a downslope (when the end point is lower than the start point).

Value

A vector of slope gradients associated with each linear element (each line between consecutive vertices) associated with linear features. Returned values for `slope_distance_mean()` and `slope_distance_mean_weighted(` are summary statistics for all linear elements in the linestring. The output value is a proportion representing the change in elevation for a given change in horizontal movement along the linestring. 0.02, for example, represents a low gradient of 2% while 0.08 represents a steep gradient of 8%.

Examples

```
x = c(0, 2, 3, 4, 5, 9)
elevations = c(1, 2, 2, 4, 3, 0) / 10 # downward slope overall
slope_vector(x, elevations)
library(sf)
m = st_coordinates(lisbon_road_segment)
d = sequential_dist(m, lonlat = FALSE)
elevations = elevation_extract(m, dem_lisbon_raster)
slope_distance(d, elevations)
slope_distance_mean(d, elevations)
slope_distance_mean(d, elevations, directed = TRUE)
slope_distance_mean(rev(d), rev(elevations), directed = TRUE)
slope_distance_weighted(d, elevations)
slope_distance_weighted(d, elevations, directed = TRUE)
```

---

slope_xyz — *Extract slopes from xyz data frame or sf objects*

Description

The function takes a sf object with 'XYZ' coordinates and returns a vector of numeric values representing the average slope of each linestring in the sf data frame input.

Usage

```
slope_xyz(
  route_xyz,
  fun = slope_matrix_weighted,
  lonlat = TRUE,
  directed = FALSE
)
```

Arguments

route_xyz	An sf or sfc object with XYZ coordinate dimensions
fun	The slope function to calculate per route, slope_matrix_weighted by default.
lonlat	Are the coordinates in lon/lat order? TRUE by default
directed	Should the value be directed? FALSE by default. If TRUE the result will be negative when it represents a downslope (when the end point is lower than the start point).

Details

The default function to calculate the mean slope is slope_matrix_weighted(). You can also use slope_matrix_mean() from the package or any other function that takes the same inputs as these functions not in the package.

Value

A vector of slopes equal in length to the number simple features (rows representing linestrings) in the input object.

Examples

```
route_xyz = lisbon_road_segment_3d
slope_xyz(route_xyz, lonlat = FALSE)
slope_xyz(route_xyz$geom, lonlat = FALSE)
slope_xyz(route_xyz, lonlat = FALSE, directed = TRUE)
slope_xyz(route_xyz, lonlat = FALSE, fun = slope_matrix_mean)
```

---

z_value *Calculate summary values for 'Z' elevation attributes*

Description

The slope_z*() functions calculate summary values for the Z axis in sfc objects with XYZ geometries.

Usage

```
z_value(x)

z_start(x)

z_end(x)

z_mean(x)

z_max(x)
```

```
z_min(x)

z_elevation_change_start_end(x)

z_direction(x)

z_cumulative_difference(x)
```

Arguments

x	An sfc object with 'XYZ' coordinates

Value

A vector of values representing elevations associated with simple feature geometries that have elevations (XYZ coordinates).

Examples

```
x = slopes::lisbon_route_3d
x
z_value(x)[1:5]
xy = slopes::lisbon_route
try(z_value(xy)) # error message
z_start(x)
z_end(x)
z_direction(x)
z_elevation_change_start_end(x)
z_direction(x)
z_cumulative_difference(x)
```

Index

* **datasets**
    cyclestreets_route, 2
    dem_lisbon_raster, 3
    lisbon_road_network, 7
    lisbon_road_segment, 8
    lisbon_route, 9
    magnolia_xy, 9

cyclestreets_route, 2

dem_lisbon_raster, 3

elevation_add, 4
elevation_extract, 5
elevation_get, 6

lisbon_road_network, 7
lisbon_road_segment, 8
lisbon_road_segment_3d (lisbon_road_segment), 8
lisbon_road_segment_xyz_mapbox (lisbon_road_segment), 8
lisbon_route, 9
lisbon_route_3d (lisbon_route), 9
lisbon_route_xyz_mapbox (lisbon_route), 9

magnolia_xy, 9

plot_dz, 10
plot_slope, 11

sequential_dist, 13
slope_distance (slope_vector), 16
slope_distance_mean (slope_vector), 16
slope_distance_weighted (slope_vector), 16
slope_matrix, 13
slope_matrix_mean (slope_matrix), 13
slope_matrix_weighted (slope_matrix), 13
slope_raster, 15
slope_vector, 16
slope_xyz, 17

z_cumulative_difference (z_value), 18
z_direction (z_value), 18
z_elevation_change_start_end (z_value), 18
z_end (z_value), 18
z_max (z_value), 18
z_mean (z_value), 18
z_min (z_value), 18
z_start (z_value), 18
z_value, 18