Package ‘elevatr’

September 12, 2023

Title Access Elevation Data from Various APIs

Version 0.99.0

URL https://github.com/jhollist/elevatr/

BugReports https://github.com/jhollist/elevatr/issues/

Maintainer Jeffrey Hollister <hollister.jeff@epa.gov>

Description Several web services are available that provide access to elevation data. This package provides access to many of those services and returns elevation data either as an 'sf' simple features object from point elevation services or as a 'raster' object from raster elevation services. In future versions, 'elevatr' will drop support for 'raster' and will instead return 'terra' objects. Currently, the package supports access to the Amazon Web Services Terrain Tiles <https://registry.opendata.aws/terrain-tiles/>, the Open Topography Global Datasets API <https://opentopography.org/developers/>, and the USGS Elevation Point Query Service <https://apps.nationalmap.gov/epqs/>.

Depends R (>= 3.5.0)

Imports httr, jsonlite, progressr, sf, terra, future, furrr, purrr, units, slippymath, curl, raster, methods

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat, knitr, rmarkdown, formatR, progress

VignetteBuilder knitr

NeedsCompilation no

Author Jeffrey Hollister [aut, cre] (<https://orcid.org/0000-0002-9254-9740>),
Tarak Shah [ctb],
Jakub Nowosad [ctb] (<https://orcid.org/0000-0002-1057-3721>),
Alec L. Robitaille [ctb] (<https://orcid.org/0000-0002-4706-1762>),
Marcus W. Beck [rev] (<https://orcid.org/0000-0002-4996-0059>),
Mike Johnson [ctb] (<https://orcid.org/0000-0002-5288-8350>)

Repository CRAN

Date/Publication 2023-09-12 21:00:02 UTC

R topics documented:

`elevatr` *Access elevation data from the web*

Description

This package provides tools to access and download elevation data available from the Mapzen elevation and Mapzen terrain service.

`get_elev_point` *Get Point Elevation*

Description

This function provides access to point elevations using either the USGS Elevation Point Query Service (US Only) or by extracting point elevations from the AWS Terrain Tiles. The function accepts a `data.frame` of x (long) and y (lat) or a `sf` `POINT` or `MULTIPOINT` object as input. A `sf` `POINT` or `MULTIPOINT` object is returned with elevation and elevation units as an added `data.frame`.

Usage

```
get_elev_point(
  locations,
  prj = NULL,
  src = c("epqs", "aws"),
  overwrite = FALSE,
  ...
)
```

Arguments

`locations`	Either a `data.frame` with x (e.g. longitude) as the first column and y (e.g. latitude) as the second column, a `SpatialPoints`/`SpatialPointsDataFrame`, or a `sf` `POINT` or `MULTIPOINT` object. Elevation for these points will be returned in the originally supplied class.
`prj`	A valid input to `st_crs`. This argument is required for a `data.frame` of locations and optional for `sf` locations.
`src`	A character indicating which API to use, either "epqs" or "aws" accepted. The "epqs" source is relatively slow for larger numbers of points (e.g. > 500). The "aws" source may be quicker in these cases provided the points are in a similar geographic area. The "aws" source downloads a DEM using `get_elev_raster` and then extracts the elevation for each point.
`overwrite`	A logical indicating that existing `elevation` and `elev_units` columns should be overwritten. Default is FALSE and `get_elev_point` will error if these columns already exist.
`...`	Additional arguments passed to get_epqs or get_aws_points. When using "aws" as the source, pay attention to the ‘z‘ argument. A defualt of 5 is used, but this uses a raster with a large ~4-5 km pixel. Additionally, the source data changes as zoom levels increase. Read `https://github.com/tilezen/joerd/blob/master/docs/data-sources.md#what-is-the-ground-resolution` for details.

Value

Function returns an `sf` object in the projection specified by the `prj` argument.

Examples

```
## Not run:
library(elevatr)
library(sf)
library(terra)

mts <- data.frame(x = c(-71.3036, -72.8145),
                  y = c(44.2700, 44.5438),
                  names = c("Mt. Washington", "Mt. Mansfield"))
ll_prj <- 4326
mts_sf <- st_as_sf(x = mts, coords = c("x", "y"), crs = ll_prj)
#Empty Raster
mts_raster <- rast(mts_sf, nrow = 5, ncol = 5)
# Raster with cells for each location
mts_raster_loc <- terra::rasterize(mts_sf, rast(mts_sf, nrow = 10, ncol = 10))

get_elev_point(locations = mts, prj = ll_prj)
get_elev_point(locations = mts, units="feet", prj = ll_prj)
get_elev_point(locations = mts_sf)
get_elev_point(locations = mts_raster)
get_elev_point(locations = mts_raster_loc)
```

```
# Code to split into a loop and grab point at a time.
# This is usually faster for points that are spread apart

library(dplyr)

elev <- vector("numeric", length = nrow(mts))
for(i in seq_along(mts)){
elev[i]<-get_elev_point(locations = mts[i,], prj = ll_prj, src = "aws",
                        z = 10)$elevation}
mts_elev <- cbind(mts, elev)
mts_elev

## End(Not run)
```

`get_elev_raster` *Get Raster Elevation*

Description

Several web services provide access to raster elevation. Currently, this function provides access to the Amazon Web Services Terrain Tiles and the Open Topography global datasets API. The function accepts a `data.frame` of x (long) and y (lat), an `sf`, or `terra` object as input. A `RasterLayer` object is returned. In subsequent versions, a `SpatRaster` will be returned.

Usage

```
get_elev_raster(
  locations,
  z,
  prj = NULL,
  src = c("aws", "gl3", "gl1", "alos", "srtm15plus"),
  expand = NULL,
  clip = c("tile", "bbox", "locations"),
  verbose = TRUE,
  neg_to_na = FALSE,
  override_size_check = FALSE,
  ...
)
```

Arguments

`locations`	Either a `data.frame` of x (long) and y (lat), an `sf`, or `terra` object as input.
`z`	The zoom level to return. The zoom ranges from 1 to 14. Resolution of the resultant raster is determined by the zoom and latitude. For details on zoom and resolution see the documentation from Mapzen at `https://github.com/tilezen/joerd/blob/master/docs/data-sources.md#what-is-the-ground-resolution`. The z is not required for the OpenTopography data sources.

`prj`	A valid input to `st_crs` If a `sf` object or a `terra` object is provided as the `locations`, the prj is optional and will be taken from `locations`. This argument is required for a `data.frame` of locations.
`src`	A character indicating which API to use. Currently supports "aws" and "gl3", "gl1", "alos", or "srtm15plus" from the OpenTopography API global datasets. "aws" is the default.
`expand`	A numeric value of a distance, in map units, used to expand the bounding box that is used to fetch the terrain tiles. This can be used for features that fall close to the edge of a tile or for retrieving additional area around the feature. If the feature is a single point, the area it returns will be small if clip is set to "bbox". Default is NULL.
`clip`	A character value used to determine clipping of returned DEM. The default value is "tile" which returns the full tiles. Other options are "bbox" which returns the DEM clipped to the bounding box of the original locations (or expanded bounding box if used), or "locations" if the spatial data (e.g. polygons) in the input locations should be used to clip the DEM. Locations are not used to clip input point datasets. Instead the bounding box is used.
`verbose`	Toggles on and off the note about units and coordinate reference system.
`neg_to_na`	Some of the data sources return large negative numbers as missing data. When the end result is a projected those large negative numbers can vary. When set to TRUE, only zero and positive values are returned. Default is FALSE.
`override_size_check`	Boolean to override size checks. Any download between 100 Mb and 500Mb report a message but continue. Between 500Mb and 3000Mb requires interaction and greater than 3000Mb fails. These can be overriden with this argument set to TRUE.
`...`	Extra arguments to pass to `httr::GET` via a named vector, `config`. See `get_aws_terrain` for more details.

Details

Currently, the `get_elev_raster` function utilizes the Amazon Web Services (`https://registry.opendata.aws/terrain-tiles/`) terrain tiles and the Open Topography Global Datasets API (`https://opentopography.org/developers`).

The AWS Terrain Tiles data is provided via x, y, and z tiles (see `https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames` for details.) The x and y are determined from the bounding box of the object submitted for `locations` argument, and the z argument must be specified by the user.

Value

Function returns a `RasterLayer` in the projection specified by the `prj` argument or in the projection of the provided locations. In subsequent versions, a `SpatRaster` will be returned.

Examples

```
## Not run:
```

```
library(elevatr)
library(sf)
data(lake)
lake_buff  <- st_buffer(lake, 1000)
loc_df <- data.frame(x = runif(6,min=sf::st_bbox(lake)$xmin,
                               max=sf::st_bbox(lake)$xmax),
                     y = runif(6,min=sf::st_bbox(lake)$ymin,
                               max=sf::st_bbox(lake)$ymax))

x <- get_elev_raster(locations = loc_df, prj = st_crs(lake) , z=10)
x <- get_elev_raster(lake, z = 12)
x <- get_elev_raster(lake, src = "gl3", expand = 5000)
x <- get_elev_raster(lake_buff, z = 10, clip = "locations")

## End(Not run)
```

`lake` *SpatialPolygonsDataFrame of Lake Sunapee*

Description

This example data is a SpatialPolygonsDataFrame of a single lake, Lake Sunapee. Used for examples and tests.

Format

SpatialPolygonDataframe with 1 lakes, each with 13 variables

`pt_df` *Small data frame of xy locations*

Description

Example data frame of locations for use in examples and text

Format

A data.frame with two columns, x(long) and y(lat) `set_opentopo_key` *Store OpenTopography Key*

Description

This function stores an OpenTopgrapy key in a local .Renviron file. If the .Renviron file exists, the key will be appended. This will typically only need to be done once per machine.

Usage

```
set_opentopo_key(key)
```

Arguments

`key`	An OpenTopography API Key as a character. For details on obtaining an Open-Topgraphy key see `https://opentopography.org/blog/introducing-api-keys-access-opentopo`

`sf_big` *A sf POINT dataset of random points*

Description

This sf POINT dataset is 250 uniform random points to be used for examples and tests

Format

A sf POINT object

Index