Package 'elevatr'

January 9, 2018

Title Access Elevation Data from Various APIs

Version 0.1.4

URL https://www.github.com/usepa/elevatr

BugReports https://github.com/usepa/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 several of those services and returns elevation data either as a SpatialPointsDataFrame from point elevation services or as a raster object from raster elevation services. Currently, the package supports access to the Mapzen Elevation Service <https://mapzen.com/documentation/elevation/elevation-service/>, Mapzen Terrain Service <https://mapzen.com/documentation/terrain-tiles/>, Amazon Web Services Terrain Tiles <https://aws.amazon.com/public-datasets/terrain/> and the USGS Elevation Point Query Service <http://ned.usgs.gov/epqs/>.

Depends R (>= 3.0.0)

Imports sp, raster, httr, jsonlite, progress, ratelimitr

License CC0

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown, formatR, rgdal

VignetteBuilder knitr

NeedsCompilation no

Author Jeffrey Hollister [aut, cre], Tarak Shah [ctb]

Repository CRAN

Date/Publication 2018-01-09 18:29:16 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

Several web services provide access to point elevations. This function provides access to several of those. Currently it uses either the Mapzen Elevation Service or the USGS Elevation Point Query Service (US Only). The function accepts a `data.frame` of x (long) and y (lat) or a `SpatialPoints/SpatialPointsDataFame` as input. A SpatialPointsDataFrame is returned with elevation as an added `data.frame`.

Usage

```
get_elev_point(locations, prj = NULL, src = c("mapzen", "epqs"),
  api_key = get_api_key(src), ...)
```

Arguments

`locations`	Either a `data.frame` with x (e.g. longitude) as the first column and y (e.g. latitude) as the second column or a `SpatialPoints/SpatialPointsDataFrame`. Elevation for these points will be returned.
`prj`	A PROJ.4 string defining the projection of the locations argument. If a `SpatialPoints` or `SpatialPointsDataFrame` is provided, the PROJ.4 string will be taken from that. This argument is required for a `data.frame` of locations.

src	A character indicating which API to use, currently "mapzen" or "epqs". Default is "mapzen". Note that the Mapzen Elevation Service is subject to rate limits. Keyless access is not allowed. With a Mapzen API key (https://mapzen.com/developers/) requests are limited to 20,000 per day or 2 per second. Per day and per second rates are not yet enforced by the elevatr package, but will be in the future. The "epqs" source is relatively slow for larger numbers of points (e.g. > 500).
api_key	A character for the appropriate API key. Default is to use key as defined in .Renviron. Acceptable environment variable name is currently only "mapzen_key" which is required. The elevatr::set_api_key function will set this key by updating the .Renviron file. An R restart is required after using elevatr::set_api_key. Defaults to Sys.getenv("mapzen_key")
...	Additional arguments passed to get_epqs or get_mapzen_elevation

Value

Function returns a SpatialPointsDataFrame in the projection specified by the prj argument.

Examples

```
## Not run:
mt_wash <- data.frame(x = -71.3036, y = 44.2700)
mt_mans <- data.frame(x = -72.8145, y = 44.5438)
mts <- rbind(mt_wash,mt_mans)
ll_prj <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
mts_sp <- sp::SpatialPoints(sp::coordinates(mts),
                            proj4string = sp::CRS(ll_prj))
get_elev_point(locations = mt_wash,prj = ll_prj)
get_elev_point(locations = mt_wash, src = "epqs", units="feet", prj = ll_prj)
get_elev_point(locations = mt_wash, src = "epqs", units="meters",
               prj = ll_prj)
get_elev_point(locations = mts_sp)
data(sp_big)
get_elev_point(sp_big)

## 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 Mapzen Terrain Service The function accepts a data.frame of x (long) and y (lat), an sp, or raster object as input. A raster object is returned.

Usage

```
get_elev_raster(locations, z, prj = NULL, src = c("mapzen", "aws"),
  api_key = get_api_key(src), expand = NULL, ...)
```

Arguments

locations	Either a data.frame of x (long) and y (lat), an sp, or raster 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://mapzen.com/documentation/terrain-tiles/data-sources/#what-is-the-ground-resolution
prj	A PROJ.4 string defining the projection of the locations argument. If a sp or raster object is provided, the PROJ.4 string will be taken from that. This argument is required for a data.frame of locations."
src	A character indicating which API to use, currently either "mapzen" (default), or "aws" is used. Both use the same source tiles. The Amazon Web Services tiles are best if rate limits are causing failure of the Mapzen tiles or if you are accessing the data via and AWS instance.
api_key	A valid API key.
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 and additional area around the feature is desired. Default is NULL.
...	Extra arguments to pass to httr::GET via a named vector, config. See get_mapzen_terrain and get_aws_terrain for more details.

Details

Currently, the get_elev_raster utilizes two separate APIs, the Mapzen Terrain Tile Service (https://mapzen.com/documentation/terrain-tiles/) or the Amazon Web Services (https://aws.amazon.com/public-datasets/terrain/). Both services utilize the same underlying data and provide global coverage, but they have different use cases. The Mapzen service is cached and thus should provide speedier downloads. It will work without an API key but an API key is suggested.

Both services are provided via x, y, and z tiles (see http://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 SpatialPointsDataFrame in the projection specified by the prj argument.

Examples

```
## Not run:
loc_df <- data.frame(x = runif(6,min=sp::bbox(lake)[1,1],
                               max=sp::bbox(lake)[1,2]),
                     y = runif(6,min=sp::bbox(lake)[2,1],
```

```
                                max=sp::bbox(lake)[2,2]))
x <- get_elev_raster(locations = loc_df, prj = sp::proj4string(lake), z=10,
                     api_key = NULL)

data(lake)
x <- get_elev_raster(lake, z = 3, src = "mapzen")
x <- get_elev_raster(lake, z = 12, src = "aws")

## 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)

---

sp_big *SpatialPoints of random points*

Description

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

Format

A SpatialPoints object

Index