Package ‘elevatr’

October 13, 2022

Title Access Elevation Data from Various APIs
Version 0.4.2
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 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
Amazon Web Services Terrain Tiles <https://registry.opendata.aws/terrain-tiles/>,
the Open Topography Global Datasets API <https://opentopography.org/developers/>,
Depends R (>= 3.5.0)
Imports sp, raster, httr, jsonlite, progressr, sf, future, furrr,
purrr, methods, units, slippymath
License CC0
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Suggests testthat, knitr, markdown, formatR, rgdal, progress
VignetteBuilder knitr
NeedsCompilation no
Author Jeffrey Hollister [aut, cre] (<https://orcid.org/0000-0002-9254-9740>),
   Tarak Shah [ctb],
   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 2022-01-07 22:12:41 UTC
get_elev_point

R topics documented:

elevatr ................................................................. 2
get_elev_point .......................................................... 2
get_elev_raster .......................................................... 4
lake ........................................................................ 6
pt_df ...................................................................... 6
set_opentopo_key ......................................................... 7
sp_big ................................................................. 7

Index 8

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 SpatialPoints/SpatialPointsDataFrame as input. A SpatialPointsDataFrame is returned with elevation as an added data.frame.

Usage

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

**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 string defining the projection of the locations argument. The string needs to be an acceptable SRS_string for CRS-class for your version of PROJ. If a sf object, a sp object or a raster object is provided, the string will be taken from that. This argument is required for a data.frame of 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 default 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 a SpatialPointsDataFrame or sf object in the projection specified by the prj argument.

**Examples**

```r
## 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 <- "EPSG:4326"
mts_sp <- sp::SpatialPoints(sp::coordinates(mts),
                          proj4string = sp::CRS(SRS_string = ll_prj))
mts_spdf <- sp::SpatialPointsDataFrame(mts_sp,
                     data = data.frame(name =
                          c("Mt. Washington", "Mt. Mansfield")))
mts_raster <- raster::raster(mts_sp, ncol = 2, nrow = 2)
get_elev_point(locations = mt_wash, prj = ll_prj)
get_elev_point(locations = mt_wash, units="feet", prj = ll_prj)
get_elev_point(locations = mt_wash, units="meters", prj = ll_prj)
get_elev_point(locations = mts_sp)
get_elev_point(locations = mts_spdf)
get_elev_point(locations = mts_raster)
```
# 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))
pb <- progress_estimated(length(elev))
for(i in seq_along(mts)){
  pb$tick()
  elev[i]<-suppressMessages(get_elev_point(locations = mts[i,], prj = ll_prj,
         src = "aws", z = 14)$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 Terrian Tiles and the Open Topography global datasets API. 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

```r
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 sp, sf, 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://github.com/tilezen/](https://github.com/tilezen/)
get_elev_raster

---

joerd/blob/master/docs/data-sources.md#what-is-the-ground-resolution.
The z is not required for the OpenTopography data sources.

prj
A string defining the projection of the locations argument. The string needs to
be an acceptable SRS_string for CRS-class for your version of PROJ. If a sf
object, a sp object or a raster object is provided, the string will be taken from
that. This argument is required for a data.frame of locations.

cmc
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 interac-
tion and greater than 3000Mb fails. These can be overridden with this argument
set to TRUE.

... Extra arguments to pass to http::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.
Examples

## Not run:
data(lake)

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]))

# Example for PROJ > 5.2.0
x <- get_elev_raster(locations = loc_df, prj = sp::wkt(lake), z=10)

# Example for PROJ < 5.2.0
x <- get_elev_raster(locations = loc_df, prj = sp::proj4string(lake), z=10)

x <- get_elev_raster(lake, z = 12)

x <- get_elev_raster(lake, src = "gl3", expand = 5000)

## 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

---

**set_opentopo_key**  
*Store OpenTopography Key*

---

**Description**

This function stores an OpenTopography 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**

```r
set_opentopo_key(key)
```

**Arguments**

- `key`: An OpenTopography API Key as a character. For details on obtaining an OpenTopography key see [https://opentopography.org/blog/introducing-api-keys-access-opentopography](https://opentopography.org/blog/introducing-api-keys-access-opentopography).

---

**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

- datasets
  - lake, 6
  - pt_df, 6
  - sp_big, 7

- elevatr, 2

- get_awsTerrain, 5
- get_elev_point, 2
- get_elev_raster, 4

- lake, 6

- pt_df, 6

- set_opentopo_key, 7
- sp_big, 7