# Package 'elevatr'

November 23, 2020

**Title** Access Elevation Data from Various APIs

**Version** 0.3.1

**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/>, and the USGS Elevation Point Query Service <https://nationalmap.gov/epqs/>.

**Depends** R (>= 3.0.0)

**Imports** sp, raster, httr, jsonlite, progress, sf, methods

**License** CC0

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Suggests** testthat, knitr, rmarkdown, formatR, rgdal

**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** 2020-11-23 14:10:06 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 `SpatialPoints/SpatialPointsDataFame` as input. A SpatialPointsDataFrame is returned with elevation as an added `data.frame`.

## Usage

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

## 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 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, 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. |

... 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 a `SpatialPointsDataFrame` or `sf` object 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, units="feet", prj = ll_prj)
get_elev_point(locations = mt_wash, units="meters", prj = ll_prj)
get_elev_point(locations = mts_sp)

# 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$tick()$print()
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

```
get_elev_raster(
  locations,
  z,
  prj = NULL,
  src = c("aws", "gl3", "gl1", "alos"),
  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`, 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/joerd/blob/master/docs/data-sources.md#what-is-the-ground-resolution. The z is not required for the OpenTopography data sources. |
| `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 supports "aws" and "gl3", "gl1", or "alos" 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 and additional area around the feature is desired. 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 spatials 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 `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)

data(lake)
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)

---

`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