Package 'ElevDistr'

April 14, 2024

Title Calculate the Distance to the Nearest Local Treeline

Version 1.0.8

Description A method to calculate the distance to the climatic tree line for large data sets of coordinates (World Geodetic System 1984) with geographical uncertainty. The default thresholds and the treeline definition is based on Paulsen and Körner (2014) <doi:10.1007/s00035-014-0124-0>, users are free to decide what climate layers they would like to use.

License MIT + file LICENSE

URL https://github.com/LivioBaetscher/ElevDistr

BugReports https://github.com/LivioBaetscher/ElevDistr/issues

Depends R (>= 3.5.0)

Imports ggmap, ggplot2, RANN, terra

Suggests knitr, rgbif, rmarkdown, testthat (>= 3.0.0), tidyverse

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Livio Bätscher [aut, cre] (<https://orcid.org/0000-0002-2989-930X>), Jurriaan M. de Vos [aut] (<https://orcid.org/0000-0001-6428-7774>)

Maintainer Livio Bätscher <livio_999@hotmail.com>

Repository CRAN

Date/Publication 2024-04-14 19:10:02 UTC

R topics documented:

`calculate_distance` *Sample and calculate the distance to the local treeline*

Description

Points are uniformly drawn along polygons that specify the treeline. The elevation of each point is then extracted and the median elevation of all points is calculated. Finally the median treeline elevation is subtracted from the `pointElevation` to get its distance to the local treeline.

Usage

```
calculate_distance(treeline, elevationRaster, pointElevation, treelineSampling = 10,
                         plot = FALSE)
```

Arguments

`treeline`	A data frame containing line-shaped polygons. Each row containing: a identifier, a start latitude and longitude, an end latitude and longitude. All longitude and latitude (WGS 84) parameters must be of the data type "numeric" and finite.
`elevationRaster`	Raster that contains a digital elevation model. Data type "SpatRaster".
`pointElevation`	Elevation of the point of interest (in meters above the sea level). One value, data type "numeric" and finite.
`treelineSampling`	A constant number of samples taken from one "treeline piece". One value, data type "integer", finite and not zero.
`plot`	Boolean that defines if a histogram of the sampled elevation is plotted. The plot will only be shown if the value is `TRUE`.

Value

Returns a numeric representing the vertical distance to the local treeline in meters.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
## Not run:
calculate_distance(treeline = dfTreeline, elevationRaster = GTOPO30, pointElevation = 512,
                   treelineSampling = 10, plot = FALSE)

## End(Not run)
```

`classify_above_treeline`

Classify points as above or below the treeline

Description

Calculates if the points (from the input data frame `coords`) are above the treeline (`TRUE`) or not (`FALSE`). This is achieved by using climate layers for growing season length and growing season temperature. For each coordinate a value from both rasters is extracted and added to the input data frame. Then points are classified, the default thresholds and the treeline definition is based on Paulsen and Körner, Alp. Bot. 124: 1-12 (2014). Classification (as boolean) is also added to the output.

Usage

```
classify_above_treeline(coords, gstRaster, gslRaster, gstTreshold = 6.4,
                                gslTreshold = 94)
```

Arguments

`coords`	Data frame representing coordinates (WGS 84) to be classified. The first column must contain the longitude, the second the latitude, both in decimal degrees. The values must be of the data type "numeric" and finite. A data frame can be generated by using the function `generate_grid`.
`gstRaster`	Climatic raster that contains the growing season temperature. Data type "SpatRaster".
`gslRaster`	Climatic raster that contains the growing season length. Data type "SpatRaster".
`gstTreshold`	Growing season temperature threshold for tree growth (in degree Celsius). One value, data type "numeric" and finite.
`gslTreshold`	Growing season length threshold for tree growth (days). One value, data type "integer" and finite.

Value

A data frame containing: longitude, latitude, growing season temperature, growing season length, and a boolean. The boolean indicates if the point is above the treeline.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Get raster layer from CHELSA
gstURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gst_1981-2010_V.2.1.tif")
gslURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gsl_1981-2010_V.2.1.tif")

gst <- terra::rast(gstURL, vsi = TRUE)
gsl <- terra::rast(gslURL, vsi = TRUE)

#Classify a single point
point <- data.frame("lon" = 8.65, "lat" = 46.87)
classify_above_treeline(coords = point, gstRaster = gst, gslRaster = gsl,
                        gstTreshold = 6.4, gslTreshold = 94)

#Classify a dummy data frame
longitude <- rep(8.53, 11)
latitude <- seq(46.8, 46.9, 0.01)
temp <- data.frame(longitude, latitude)
classify_above_treeline(coords = temp, gstRaster = gst, gslRaster = gsl,
                        gstTreshold = 6.4, gslTreshold = 94)
```

---

distance_to_treeline *Wrapper that calculates the distance relative to the nearest local treeline*

Description

Calculate the distance to the treeline in meters. Positive values indicate that the sample is above the treeline. Negative values for samples below the treeline.

Usage

```
distance_to_treeline(lon, lat, gstRaster, gslRaster, elevationRaster, elevation,
                     pointDf , gridSize = 10,  gridStepSize = 0.0025, plot = FALSE,
                     plotZoom = NULL, treelineSamplingSize = 10, plotHist = FALSE,
                             gstMin = 6.4, gslMin = 94)
```

Arguments

lon	Longitude of a point (in degrees; WGS 84). One value or a vector, data type "numeric" and finite.
lat	Latitude of a point (in degrees; WGS 84). One value or a vector, data type "numeric" and finite.
gstRaster	Climatic raster that contains the growing season temperature. Data type "SpatRaster".
gslRaster	Climatic raster that contains the growing season length. Data type "SpatRaster".

`elevationRaster`
: Raster that contains a digital elevation model. Data type "SpatRaster".

`elevation`
: Elevation of the point of interest (in meters above the sea level). One value or a vector, data type "numeric" and finite.

`pointDf`
: Data frame that contains coordinates (WGS 84) of points above the treeline. The first column must contain the longitude, the second the latitude. The values must be of the data type "numeric" and finite.

`gridSize`
: Square size (in km) of the grid. One value, data type "numeric" and finite.

`gridStepSize`
: Step size for the square sampling (in degree) of the grid. One value, data type "numeric" and finite.

`plot`
: Boolean that defines if a map of the sampled area is plotted. The plot will only be shown if the value is `TRUE`.

`plotZoom`
: Map zoom, for the "get_map" function of the "ggmap" library. One value, data type "integer", from 3 to 21 and finite.

`treelineSamplingSize`
: A constant number of samples taken from one "treeline piece". One value, data type "integer", not zero and finite.

`plotHist`
: Boolean that defines if a histogram of the sampled elevation is plotted. The plot will only be shown if the value is `TRUE`.

`gstMin`
: Growing season temperature threshold for tree growth (in degree Celsius). One value, data type "numeric" and finite.

`gslMin`
: Growing season length threshold for tree growth (days). One value, data type "numeric" and finite.

Details

This is the main function, which calls the other relevant functions. Specifically, in turn, it calls `get_nearest_point` to identify where the nearest local treeline is, `generate_grid`, `classify_above_treeline`, and `sample_treeline` to locally investigate at what elevation the treeline is, and finally `calculate_distance` to determine the elevation of the point relative to the local treeline. It is recommended to use this wrapper rather than the individual functions, unless you have a very specific reason not to. The position of a point relative to the treeline depends on a treeline definition. Here we follow the definition of Paulsen & Körner, Alp. Bot. 124: 1-12 (2014), which is based on specific thresholds of growing season length and growing season temperature (94 days and 9.4°C, respectively). It is possible to adjust these thresholds manually, e.g. to achieve a elevation above or below another climatic line. Note that this requires to first calculate `pointDf` for the boundary of interest using the functions `generate_point_df`. Because the implemented treeline definition depends not only on temperature but also on growing season length, it can be affected by drought. Therefore, the user must take care in interpreting treeline information in desert mountain systems. Here, we recommend to frequently use the option `plot` and `plotHist` to gain a thorough understanding of the local situation.

Value

Returns the distance to the local treeline in meters as one value or as vector.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Get raster layer from CHELSA
gstURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gst_1981-2010_V.2.1.tif")
gslURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gsl_1981-2010_V.2.1.tif")

gst <- terra::rast(gstURL, vsi = TRUE)
gsl <- terra::rast(gslURL, vsi = TRUE)

gmted2010URL <- paste0("https://edcintl.cr.usgs.gov/downloads/sciweb1/shared/topo/downloads/GMTED/",
               "Global_tiles_GMTED/300darcsec/med/E000/30N000E_20101117_gmted_med300.tif")
gmted2010Part <- terra::rast(gmted2010URL, vsi = TRUE)

#Check one point
distance_to_treeline(lon = 8.65, lat = 46.87, gstRaster = gst, gslRaster = gsl,
                     elevationRaster = gmted2010Part, elevation = 504,
                     pointDf = pointsAboveTreeline, plot = FALSE,
                     plotHist = FALSE, gstMin = 6.4, gslMin = 94)
distance_to_treeline(lon = 4.47, lat = 51.92, gstRaster = gst, gslRaster = gsl,
                     elevationRaster = gmted2010Part, elevation = 504,
                     pointDf = pointsAboveTreeline, plot = FALSE,
                     plotHist = FALSE, gstMin = 6.4, gslMin = 94)
distance_to_treeline(lon = -156.71, lat = 69.74,gstRaster = gst, gslRaster = gsl,
                     elevationRaster = gmted2010Part, elevation = 504,
                     pointDf = pointsAboveTreeline, plot = FALSE,
                     plotHist = FALSE, gstMin = 6.4, gslMin = 94)
```

---

generate_grid *Generate a grid around the input point*

Description

Generate a grid around a longitude and latitude, with a defined square size and step size.

Usage

```
generate_grid(lon, lat, squareSize = 10, stepSize = 0.0025)
```

Arguments

lon	Longitude of the grid center (in degrees; WGS 84). One value, data type "numeric" and finite.

lat	Latitude of the grid center (in degrees; WGS 84). One value, data type "numeric" and finite.
squareSize	Square size (in km). One value, data type "numeric" and finite.
stepSize	Step size for the square sampling (in degree). One value, data type "numeric" and finite.

Value

A list containing a data frame with longitude and latitude of the grid and a vector containing the length of the longitudinal and latitudinal sequence.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Generate a 10x10 km grid with a step size of 0.0025 degrees
temp <- generate_grid(lon = 8.728898, lat = 46.93756, squareSize = 10, stepSize = 0.0025)

#Part of the generated coordinates
temp$df[105:115,]
```

---

generate_point_df *Generate a data frame with points above the treeline*

Description

A data frame is generated containing only points that are at or above the treeline. The calculation of the treeline (when using default thresholds) is based on Paulsen and Körner, Alp. Bot. 124: 1-12 (2014).

Usage

```
generate_point_df(gstRaster, gslRaster, stepSize = 0.0416666,
                        gstTreshold = 6.4, gslTreshold = 94)
```

Arguments

gstRaster	Climatic raster that contains the growing season temperature. Data type "SpatRaster".
gslRaster	Climatic raster that contains the growing season length. Data type "SpatRaster".
stepSize	Step size for the sampling (in degree). This defines how fare the coordinates are apart. One value, data type "numeric" and finite.
gstTreshold	Growing season temperature threshold for tree growth (in degree Celsius). One value, data type "numeric" and finite.
gslTreshold	Growing season length threshold for tree growth (days). One value, data type "integer" and finite.

Value

Data frame that contains coordinates of points above the treeline.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Get raster layer from CHELSA
gstURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gst_1981-2010_V.2.1.tif")
gslURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gsl_1981-2010_V.2.1.tif")

gst <- terra::rast(gstURL, vsi = TRUE)
gsl <- terra::rast(gslURL, vsi = TRUE)

#Now generate a example data frame
temp <- generate_point_df(gstRaster = gst, gslRaster = gsl, stepSize = 10,
                          gstTreshold = 6.4, gslTreshold = 94)
```

---

get_nearest_point *Search the nearest point in a data frame*

Description

Search for the nearest point in a data frame using a k-dimensional tree and a nearest neighbor search. The aim of this function is to get the nearest point above the treeline given the chosen `lon` and `lat`.

Usage

```
get_nearest_point(lon, lat, pointDf)
```

Arguments

`lon`	Longitude of a point (in degrees; WGS 84). One value, data type "numeric" from -180 until 180 and finite.
`lat`	Latitude of a point (in degrees; WGS 84). One value, data type "numeric" from -90 until 90 and finite.
`pointDf`	Data frame that contains coordinates (WGS 84) of points above the treeline. The first column must contain the longitude, the second the latitude. The values must be of the data type "numeric" and finite.

Value

A list containing the longitude and the latitude of the nearest point.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Create a dummy data frame.
longitude <- seq(0, 10)
latitude <- seq(40, 50)
temp <- data.frame(longitude, latitude)
get_nearest_point(lon = 8.65, lat = 46.87, pointDf = temp)

#Use the data that is included in the package.
get_nearest_point(lon = 8.65, lat = 46.87, pointDf = pointsAboveTreeline)
```

---

plot_distr *Plot the sampled area*

Description

With this function it is possible to plot the analyzed area. However you need to register a APIs. If you are not willing to do this, you cannot plot and the function will throw a warning. See: https://www.rdocumentation.org/packages/ggmap/versions/3.0.0.

Usage

```
plot_distr(nearestCorner, grid, treelineDf, size = 12)
```

Arguments

nearestCorner	A list containing the longitude and the latitude (WGS 84) of the point which is used to load the map. The values must be of the data type "numeric" and finite.
grid	Data frame generated by the function classify_above_treeline and therefore containing: longitude, latitude (WGS 84), growing season temperature, growing season length, and a boolean. Longitude and latitude must be of the data type "numeric" and finite. For the boolean TRUE, FALSE and NA is allowed and nothing else.
treelineDf	A data frame containing line-shaped polygons. Each row containing: a identifier, a start latitude and longitude, a end latitude and longitude (all WGS 84). All longitude and latitude parameters must be of the type "numeric" and finite.
size	Map zoom, for the "get_map" function of the "ggmap" library. One value, data type "integer", finite and in the range from 3 to 21.

Value

Nothing.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
## Not run:
plot_distr(nearestCorner = pointsAboveTreeLine, grid = dfGrid, treelineDf = dfTreeline,
           size = 12)

## End(Not run)
```

`pointsAboveTreeline` *Points on or above the treeline*

Description

A data set containing 133,302 points on or above the treeline. The data set was generated with the function `generate_point_df`. For the `gstRaster` (growing season temperature) and the `gslRaster` (growing season length) raster layers from CHELSA V2.1 are used (see source), the `stepSize` is set to 0.0416666 degrees. The thresholds and the treeline definition is based on Paulsen and Körner, Alp. Bot. 124: 1-12 (2014). The GMBA mountain inventory V1.2 was used to remove points that do not lie within steep terrain.

Usage

```
pointsAboveTreeline
```

Format

A data frame with 511'930 rows and 2 variables:

longitude Longitude of the point (in degrees; WGS 84).

latitude Latitude of the point (in degrees; WGS 84).

Source

gstRaster: https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/GLOBAL/climatologies/1981-2010/bio/CHELSA_gst_1981-2010_V.2.1.tif

gslRaster: https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/GLOBAL/climatologies/1981-2010/bio/CHELSA_gsl_1981-2010_V.2.1.tif

`sample_treeline` *Compute the local treeline*

Description

Calculate horizontal and vertical lines between two different classified points from the `df` input. If used in the context of the treeline: when a point above the treeline (`TRUE`) and a point below the treeline (`FALSE`) lie next to each other, the start and the end of the line is calculated and stored. This data point collection represents the local treeline. It is highly recommended to use this function only in combination with `generate_grid` and `classify_above_treeline`. The coordinates in the `df` can only be meaningfully processed if they have the same order and structure as results from `generate_grid`.

Usage

```
sample_treeline(df, lonLength, latLength, stepSize = 0.0025)
```

Arguments

`df`	Data frame generated by the function `classify_above_treeline` and therefore containing: longitude, latitude (WGS 84), growing season temperature, growing season length, and a boolean. Longitude and latitude must be of the data type "numeric" and finite. For the boolean `TRUE`, `FALSE` and `NA` is allowed and nothing else.
`lonLength`	Vector containing the length of the longitudinal sequence. One value, data type "numeric". This information is part of the `generate_grid` output. One value, data type "numeric" and finite.
`latLength`	Vector containing the length of the latitudinal sequence. One value, data type "numeric". This information is part of the `generate_grid` output. One value, data type "numeric" and finite.
`stepSize`	Step size for the square sampling (in degree). One value, data type "numeric". This `stepSize` must be identical with the `stepSize` used in the function `generate_grid`. It is used to calculate the center between two grid points. One value, data type "numeric" and finite.

Value

A data frame containing line-shaped polygons. Each row containing: a identifier, a start latitude and longitude, a end latitude and longitude.

Author(s)

Livio Bätscher, Jurriaan M. de Vos

Examples

```
#Recommended usage
temp <- generate_grid(lon = 8.728898, lat = 46.93756, squareSize = 10, stepSize = 0.0025)

gstURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gst_1981-2010_V.2.1.tif")
gslURL <- paste0("https://os.zhdk.cloud.switch.ch/envicloud/chelsa/chelsa_V2/",
                 "GLOBAL/climatologies/1981-2010/bio/CHELSA_gsl_1981-2010_V.2.1.tif")

gst <- terra::rast(gstURL, vsi = TRUE)
gsl <- terra::rast(gslURL, vsi = TRUE)

temp$df <- classify_above_treeline(coords = temp$df, gstRaster = gst, gslRaster = gsl)

treeline <- sample_treeline(df = temp$df, lonLength = temp$lonLength,
                            latLength = temp$latLength, stepSize = 0.0025)
```

Index