Package 'BIOMASS'

March 11, 2026

Type Package

Title Estimating Aboveground Biomass and Its Uncertainty in Tropical Forests

Version 2.2.7

Date 2026-03-10

Description Contains functions for estimating above-ground biomass/carbon and its uncertainty in tropical forests. These functions allow to (1) retrieve and correct taxonomy, (2) estimate wood density and its uncertainty, (3) build height-diameter models, (4) manage tree and plot coordinates, (5) estimate above-ground biomass/carbon at stand level with associated uncertainty. To cite 'BIOMASS', please use citation('BIOMASS'). For more information, see Réjou-Méchain et al. (2017) <doi:10.1111/2041-210X.12753>.

License GPL-2

Encoding UTF-8

LazyData true

Depends R(>= 3.6)

URL https://umr-amap.github.io/BIOMASS/, https://github.com/umr-amap/BIOMASS/

BugReports https://github.com/umr-amap/BIOMASS/issues/

Imports minpack.lm, jsonlite, methods, proj4, graphics, stats, utils, data.table (>= 1.9.8), rappdirs, sf, terra, ggplot2

VignetteBuilder knitr

Suggests knitr, rmarkdown, prettydoc, testthat, vdiffr, curl, geodata, httr2, pkgdown, dplyr, brms, BH, RcppEigen

RoxygenNote 7.3.3

NeedsCompilation no

Author Dominique Lamonica [aut, cre], Maxime Réjou-Méchain [aut, dtc], Arthur Bailly [aut], Guillaume Cornu [aut] (ORCID: <https://orcid.org/0000-0002-7523-5176>), John Godlee [ctb],

Fabian Fischer [ctb],
Jerome Chave [ctb],
Arthur Pere [aut],
Ariane Tanguy [aut],
Camille Piponiot [aut],
Bruno Hérault [aut],
Philippe Verley [ctb],
Ted Feldpausch [dtc]

Maintainer Dominique Lamonica <dominique.lamonica@ird.fr>

Repository CRAN

Date/Publication 2026-03-11 06:30:37 UTC

Contents

`AGBmonteCarlo` *Propagating above ground biomass (AGB) or carbon (AGC) errors to the stand level*

Description

Propagation of the errors throughout the steps needed to compute AGB or AGC.

Usage

```
AGBmonteCarlo(
  D,
  WD = NULL,
  errWD = NULL,
  H = NULL,
  errH = NULL,
  HDmodel = NULL,
  coord = NULL,
  Dpropag = NULL,
  n = 1000,
  Carbon = FALSE,
  Dlim = NULL
)
```

Arguments

`D`	Vector of tree diameters (in cm)
`WD`	Vector of wood density estimates (in g/cm3)
`errWD`	Vector of error associated to the wood density estimates (should be of the same size as `WD`)
`H`	(option 1) Vector of tree heights (in m). If set, `errH` must be set too.
`errH`	(if `H`) Residual standard error (RSE) of a model or vector of errors (sd values) associated to tree height values (in the latter case the vector should be of the same length as `H`).
`HDmodel`	(option 2) Model used to estimate tree height from tree diameter (output from `modelHD()`, see example).
`coord`	(option 3) Coordinates of the site(s), either a vector giving a single site (e.g. c(longitude, latitude)) or a matrix/dataframe with two columns (e.g. cbind(longitude, latitude)). The coordinates are used to predict height-diameter allometry with bioclimatic variables.
`Dpropag`	This variable can take three kind of values, indicating how to propagate the errors on diameter measurements: a single numerical value or a vector of the same size as `D`, both representing the standard deviation associated with the diameter measurements or `"chave2004"` (an important error on 5 percent of the measures, a smaller error on 95 percent of the trees).

n	Number of iterations. Cannot be smaller than 50 or larger than 1000. By default n = 1000
Carbon	(logical) Whether or not the propagation should be done up to the carbon value (FALSE by default).
Dlim	(optional) Minimum diameter (in cm) for which above ground biomass should be calculated (all diameter below Dlim will have a 0 value in the output).

Details

See Rejou-Mechain et al. (2017) for all details on the error propagation procedure.

Value

Returns a list with (if Carbon is FALSE):

- meanAGB: Mean stand AGB value following the error propagation
- medAGB: Median stand AGB value following the error propagation
- sdAGB: Standard deviation of the stand AGB value following the error propagation
- credibilityAGB: Credibility interval at 95\
- AGB_simu: Matrix with the AGB of the trees (rows) times the n iterations (columns)

Author(s)

Maxime REJOU-MECHAIN, Bruno HERAULT, Camille PIPONIOT, Ariane TANGUY, Arthur PERE

References

Chave, J. et al. (2004). *Error propagation and scaling for tropical forest biomass estimates*. Philosophical Transactions of the Royal Society B: Biological Sciences, 359(1443), 409-420.

Rejou-Mechain et al. (2017). *BIOMASS: An R Package for estimating above-ground biomass and its uncertainty in tropical forests*. Methods in Ecology and Evolution, 8 (9), 1163-1167.

Examples

```
# Load a database
data(NouraguesHD)
data(NouraguesTrees)

# Modelling height-diameter relationship
HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H, method = "log2")

# Retrieving wood density values

NouraguesWD <- getWoodDensity(NouraguesTrees$Genus, NouraguesTrees$Species,
  stand = NouraguesTrees$Plot
)
```

```
# Propagating errors with a standard error for Wood density

resultMC <- AGBmonteCarlo(
  D = NouraguesTrees$D, WD = NouraguesWD$meanWD,
  errWD = NouraguesWD$sdWD, HDmodel = HDmodel
)


# If only the coordinates are available
coord <- c(-52.683213,4.083024 )

resultMC <- AGBmonteCarlo(
  D = NouraguesTrees$D, WD = NouraguesWD$meanWD,
  errWD = NouraguesWD$sdWD, coord = coord
)


# Propagating errors with a standard error in wood density in all plots at once

NouraguesTrees$meanWD <- NouraguesWD$meanWD
NouraguesTrees$sdWD <- NouraguesWD$sdWD
resultMC <- by(
  NouraguesTrees, NouraguesTrees$Plot,
  function(x) AGBmonteCarlo(
      D = x$D, WD = x$meanWD, errWD = x$sdWD,
      HDmodel = HDmodel, Dpropag = "chave2004"
    )
)
meanAGBperplot <- unlist(sapply(resultMC, "[", 1))
credperplot <- sapply(resultMC, "[", 4)
```

`attributeTree` *Attribute trees to subplots*

Description

Function to attribute the trees on each subplot, the trees that are at the exterior of the subplot will be marked as NA

Usage

```
attributeTree(xy, plot, coordAbs)
```

Arguments

`xy`	The coordinates of the trees for each plot
`plot`	The label of the plot (same length as the number of rows of `xy`)
`coordAbs`	Output of the function `cutPlot()`

Value

A vector with the code of the subplot for each trees, the code will be `plot_X_Y`. `X` and `Y` are the coordinate where the tree is inside the plot in regards to the corresponding subplot.

Author(s)

Arthur PERE

Examples

```
# Trees relative coordinates
xy <- data.frame(x = runif(200, min = 0, max = 200), y = runif(200, min = 0, max = 200))


# cut the plot in multiple part
coord <- data.frame(X = rep(c(0, 200, 0, 200), 2), Y = rep(c(0, 0, 200, 200), 2))
coord[1:4, ] <- coord[1:4, ] + 5000
coord[5:8, ] <- coord[5:8, ] + 6000
corner <- rep(c(1, 2, 4, 3), 2)
plot <- rep(c("plot1", "plot2"), each = 4)

cut <- cutPlot(coord, plot, corner, gridsize = 100, dimX = 200, dimY = 200)


# Assign a plot to 200 trees
plot <- rep(c("plot1", "plot2"), 100)

# attribute trees to subplots
attributeTree(xy, plot, cut)
```

---

attributeTreeCoord *Attribute GPS coordinates to trees*

Description

Attribute GPS coordinates to trees

Usage

```
attributeTreeCoord(xy, plot, dim, coordAbs)
```

Arguments

`xy`	The relative coordinates of the trees within each plot
`plot`	The label of the plot (same length as the number of rows of `xy` or length of 1)
`dim`	The dimension of the plot (either one value if the plot is a square or a vector if a rectangle)
`coordAbs`	The result of the function `cutPlot()` or `numberCorner()`

Value

A data frame with two columns: - `Xproj`: The `X` coordinates in the absolute coordinate system - `Yproj`: The `Y` coordinates in the absolute coordinate system

Examples

```
# Trees relative coordinates
xy <- data.frame(x = runif(200, min = 0, max = 200), y = runif(200, min = 0, max = 200))


# cut the plot in multiple part
coord <- data.frame(X = rep(c(0, 200, 0, 200), 2), Y = rep(c(0, 0, 200, 200), 2))
coord[1:4, ] <- coord[1:4, ] + 5000
coord[5:8, ] <- coord[5:8, ] + 6000
corner <- rep(c(1, 2, 4, 3), 2)
Forestplot <- rep(c("plot1", "plot2"), each = 4)

Outcut <- cutPlot(coord, Forestplot, corner, gridsize = 100, dimX = 200, dimY = 200)


# Assign a plot to 200 trees
Forestplot <- rep(c("plot1", "plot2"), 100)

# attribute trees to subplots
attributeTreeCoord(xy, Forestplot, dim =100,coordAbs = Outcut)
```

---

bilinear_interpolation

Generalized bilinear interpolation of coordinates

Description

Apply a generalized bilinear interpolation to convert any coordinates from one original coordinate system to another, using the plot's 4 corner coordinates of both system.

Usage

```
bilinear_interpolation(
  coord,
  from_corner_coord,
  to_corner_coord,
  ordered_corner = F
)
```

Arguments

`coord`	a matrix or data.frame : coordinates to be transformed, with X and Y corresponding to the first two columns

`from_corner_coord`
: a matrix or data.frame : corner coordinates of the rectangular plot in the original coordinate system, with X and Y corresponding to the first two columns

`to_corner_coord`
: a matrix or data.frame : corner coordinates of the plot in the coordinate system to be projected, with the same line order as from_corner_coord and , with X and Y corresponding to the first two columns

`ordered_corner`
: a logical, if TRUE : indicating that from_corner_coord and to_corner_coord rows are sorted in correct order (clockwise or counter-clockwise)

Details

The plot represented by the 4 coordinates in from_corner_coord must have 4 right angles, i.e. a rectangular (or square) plot.

When ordered_corner = FALSE, the function automatically reassigns corners in a counter-clockwise order.

Value

a data.frame containing the converted coordinates

Author(s)

Arthur Bailly

References

C. -C. Wei and C. -H. Chen, "Generalized Bilinear Interpolation of Motion Vectors for Quad-Tree Mesh," 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, China, 2008, pp. 635-638, doi: 10.1109/IIH-MSP.2008.283.

Examples

```
from_corner_coord <- expand.grid(X = c(0, 100), Y = c(0, 50))
rot_mat <- matrix(c(cos(-pi/6),sin(-pi/6),-sin(-pi/6),cos(-pi/6)),nrow=2)
to_corner_coord <- as.matrix(from_corner_coord) %*% rot_mat
to_corner_coord <- sweep(to_corner_coord, 2, c(50,100), FUN = "+")
coord <- expand.grid(X = seq(0,100,10), Y = seq(0,50,5))
projCoord = bilinear_interpolation(coord = coord,
                                   from_corner_coord = from_corner_coord,
                                   to_corner_coord = to_corner_coord)
```

`cacheManager` *Function that return a possibly cached file, transparently downloading it if missing*

Description

Function that return a possibly cached file, transparently downloading it if missing

Usage

```
cacheManager(nameFile)
```

Arguments

`nameFile`	character. file to resolve cached path.

Value

file path of the resolved cached file.

Localisation

Cache path discovery protocol

1. BIOMASS.cache option set to an **existing** folder
2. **existing** user data folder `rappdirs::user_data_dir()`
   - On Linux : `~/.local/share/R/BIOMASS`
   - On Mac OS X : `~/Library/Application Support/R/BIOMASS`
   - On Windows 7 up to 10 : `C:\\Users\\<username>\\AppData\\Local\\R\\BIOMASS`
   - On Windows XP : `C:\\Documents and Settings\\<username>\\Data\\R\\BIOMASS`
3. fallback to R session tempdir

`cachePath` *Function used to build a file path based on a cache folder*

Description

Parameters are similar to that of file.path function

Usage

```
cachePath(...)
```

Arguments

`...`	character vectors. Elements of the subpath of cache path

Value

A character vector of normalized file path with a source attribute holding a hint to cache path source ("option", "data", "temp")

Localisation

Cache path discovery protocol

1. BIOMASS.cache option set to an **existing** folder
2. **existing** user data folder `rappdirs::user_data_dir()`
   - On Linux : `~/.local/share/R/BIOMASS`
   - On Mac OS X : `~/Library/Application Support/R/BIOMASS`
   - On Windows 7 up to 10 : `C:\\Users\\<username>\\AppData\\Local\\R\\BIOMASS`
   - On Windows XP : `C:\\Documents and Settings\\<username>\\Data\\R\\BIOMASS`
3. fallback to R session tempdir

---

check_plot_coord *Check coordinates of plot corners and trees*

Description

Quality check of plot corner and tree coordinates.

Usage

```
check_plot_coord(
  corner_data,
  proj_coord = NULL,
  longlat = NULL,
  rel_coord,
  trust_GPS_corners,
  draw_plot = TRUE,
  tree_data = NULL,
  tree_coords = NULL,
  max_dist = 10,
  rm_outliers = TRUE,
  plot_ID = NULL,
  tree_plot_ID = NULL,
  ref_raster = NULL,
  shapefile = NULL,
  prop_tree = NULL,
  threshold_tree = NULL,
  ask = TRUE
)
```

Arguments

corner_data	A data frame, data frame extension, containing the plot corner coordinates.
proj_coord	(optional, if longlat is not provided) A character vector of length 2, specifying the column names (resp. x, y) of the corner projected coordinates.
longlat	(optional, if proj_coord is not provided) A character vector of length 2 specifying the column names of the corner geographic coordinates (long,lat).
rel_coord	A character vector of length 2 specifying the column names (resp. x, y) of the corner relative coordinates (that of the field, ie, the local ones).
trust_GPS_corners	A logical indicating whether or not you trust the GPS coordinates of the plot's corners. See details.
draw_plot	A logical indicating if the plot design should be displayed and returned.
tree_data	A data frame, data frame extension, containing the relative coordinates (field/local coordinates) of the trees and optional other tree metrics.
tree_coords	A character vector specifying the column names of the tree relative coordinates.
max_dist	If dealing with repeated measurements of each corner : the maximum distance (in meters) above which GPS measurements should be considered outliers (default 15 m).
rm_outliers	If TRUE and dealing with repeated measurements of each corner, then outliers are removed from the coordinate calculation of the referenced corners.
plot_ID	If dealing with multiple plots : a character indicating the variable name for corner plot IDs in corner_data.
tree_plot_ID	If dealing with multiple plots : a character indicating the variable name for tree plot IDs in tree_data.
ref_raster	filename (character) of the raster to be displayed (typically a CHM raster created from LiDAR data), or a SpatRaster object from terra package.
shapefile	filename (character) of the shapefile to be displayed, or an object of class 'sf' (sf package).
prop_tree	The column name variable of tree_data for which the tree visualization will be proportional.
threshold_tree	a numeric of length 1: the threshold of the 'prop_tree' variable at which trees will be displayed on the plot.
ask	If TRUE and dealing with multiple plots, then prompt user before displaying each plot.

Details

If trust_GPS_corners is TRUE, corner coordinates in the projected coordinate system are averaging by corner (if multiple measures) and outlier corners are identified sequentially using these averages and the max_dist argument. Then, projected coordinates of the trees are calculated from the local coordinates using a bilinear interpolation that follows the correspondence of the corners between these two coordinate systems. Be aware that this projection only works if the plot, in the relative coordinates system, is rectangular (ie, has 4 right angles).

If trust_GPS_corners is FALSE, corner coordinates in the projected coordinate system are calculated by a procrust analysis that preserves the shape and dimensions of the plot in the local coordinate system. Outlier corners are also identified sequentially and projected coordinates of the trees are calculated by applying the resulting procrust analysis.

If longlat is provided instead of proj_coord, the function will first convert the long/lat coordinates into UTM coordinates. An error may result if the parcel is located right between two UTM zones. In this case, the user has to convert himself his long/lat coordinates into any projected coordinates which have the same dimension than his local coordinates (in meters most of the time).

If longlat and proj_coord are provided, only longitude/latitude coordinates will be considered.

When ref_raster is provided, this raster is cropped for every plot contained in corner_data.

Value

Returns a list including :

- `corner_coord`: a data frame containing the projected coordinates (x_proj and y_proj) and the relative coordinates (x_rel and y_rel) of the 4 corners of the plot
- `polygon`: a sf object containing plot's polygon(s)
- `tree_data`: if `tree_data` is provided in the arguments of the function, a data frame corresponding to tree_data for which the projected coordinates of the trees (x_proj and y_proj) are added, and also a variable telling if the trees are inside the plot (is_in_plot). The name of the relative tree coordinates are also standardised and renamed to (x_rel and y_rel).
- `outliers`: a data frame containing the projected coordinates and the row number of GPS measurements considered outliers
- `plot_design`: if `draw_plot` is TRUE, a ggplot object corresponding to the design of the plot
- `UTM_code`: if `longlat` is provided, a data.frame containing the UTM code of the corner GPS coordinates for each plot
- `sd_coord`: a data frame containing (for each plot) the average standard deviation of the GPS measurements for each corner on the X and Y axes.

Author(s)

Arthur PERE, Maxime REJOU-MECHAIN, Arthur BAILLY

Arthur BAILLY, Arthur PERE, Maxime REJOU-MECHAIN

Examples

```
# One plot with repeated measurements of each corner
data("NouraguesPlot201")
check_plot201 <- check_plot_coord(
  corner_data = NouraguesPlot201,
  proj_coord = c("Xutm","Yutm"), rel_coord = c("Xfield","Yfield"),
  trust_GPS_corners = TRUE, draw_plot = FALSE)
check_plot201$corner_coord

  check_plot201$plot_design
```

```
# 4 plots with one measurement of each corner
data("NouraguesCoords")
check_plots <- check_plot_coord(
  corner_data = NouraguesCoords,
  proj_coord = c("Xutm","Yutm"), rel_coord = c("Xfield","Yfield"),
  trust_GPS_corners = TRUE, plot_ID = "Plot", draw_plot = FALSE)
check_plots$corner_coord

  check_plots$plot_design


# Displaying the associated CHM raster and representing trees proportionally to their diameter
plot_204_coords <- NouraguesCoords[NouraguesCoords$Plot==204,]
data("NouraguesTrees")
plot_204_trees <- NouraguesTrees[NouraguesTrees$Plot == 204, ]
nouragues_raster <- terra::rast(
  system.file("extdata", "NouraguesRaster.tif",
              package = "BIOMASS", mustWork = TRUE)
  )
check_plot_204 <- check_plot_coord(
 corner_data = plot_204_coords,
 proj_coord = c("Xutm","Yutm"), rel_coord = c("Xfield","Yfield"),
 trust_GPS_corners = TRUE, draw_plot = FALSE,
 tree_data = plot_204_trees, tree_coords = c("Xfield","Yfield"),
 ref_raster = nouragues_raster, prop_tree = "D", threshold_tree = 25
)

  check_plot_204$plot_design
```

---

clearCache *Function to clear cache content and possibly remove it*

Description

It will refuse to clear or remove a custom cache folder set using BIOMASS.cache option as we don't know whether this folder contains other possibly valuable files apart from our cached files.

Usage

```
clearCache(remove = FALSE)
```

Arguments

remove	logical. If TRUE cache folder will be removed too (not only content) resulting in deactivating cache as a side effect

Value

No return value, called for side effects

computeAGB *Computing tree above ground biomass (AGB)*

Description

This function uses Chave et al. 2014's pantropical models to estimate the above ground biomass of tropical trees.

Usage

```
computeAGB(D, WD, H = NULL, coord = NULL, Dlim = NULL)
```

Arguments

D	Tree diameter (in cm), either a vector or a single value.
WD	Wood density (in g/cm3), either a vector or a single value. If not available, see getWoodDensity().
H	(optional) Tree height (H in m), either a vector or a single value. If not available, see retrieveH() and modelHD(). Compulsory if the coordinates coord are not given.
coord	(optional) Coordinates of the site(s), either a vector giving a single site (e.g. c(longitude, latitude)) or a matrix/dataframe with two columns (e.g. cbind(longitude, latitude)). The coordinates are used to account for variation in height-diameter relationship thanks to an environmental proxy (parameter E in Chave et al. 2014). Compulsory if tree heights H are not given.
Dlim	(optional) Minimum diameter (in cm) for which aboveground biomass should be calculated (all diameter below Dlim will have a 0 value in the output).

Details

This function uses two different ways of computing the above ground biomass of a tree:

1. If tree height data are available, the AGB is computed thanks to the following equation (Eq. 4 in Chave et al., 2014):

$$AGB = 0.0673 * (WD * H * D^2)^{0}.976$$

2. If no tree height data is available, the AGB is computed thanks to the site coordinates with the following equation, slightly modified from Eq. 7 in Chave et al., 2014 (see Réjou-Méchain et al. 2017):

$$AGB = exp(-2.024 - 0.896 * E + 0.920 * log(WD) + 2.795 * log(D) - 0.0461 * (log(D)^2))$$

where E is a measure of environmental stress estimated from the site coordinates (coord).

Value

The function returns the AGB in Mg (or ton) as a single value or a vector.

Author(s)

Maxime REJOU-MECHAIN, Ariane TANGUY, Arthur PERE

References

Chave et al. (2014) *Improved allometric models to estimate the aboveground biomass of tropical trees*, Global Change Biology, 20 (10), 3177-3190

See Also

computeE()

Examples

```
# Create variables
D <- 10:99
WD <- runif(length(D), min = 0.1, max = 1)
H <- D^(2 / 3)

# If you have height data
AGB <- computeAGB(D, WD, H)

# If you do not have height data and a single site
lat <- 4.08
long <- -52.68
coord <- c(long, lat)

AGB <- computeAGB(D, WD, coord = coord)


# If you do not have height data and several sites (here three)
lat <- c(rep(4.08, 30), rep(3.98, 30), rep(4.12, 30))
long <- c(rep(-52.68, 30), rep(-53.12, 30), rep(-53.29, 30))
coord <- cbind(long, lat)

AGB <- computeAGB(D, WD, coord = coord)
```

---

computeFeldRegion *Retrieving Feldpausch regions*

Description

Extract the Feldpausch et al. (2012)'s regions using local coordinates.

Usage

```
computeFeldRegion(coord, level = c("region"))
```

Arguments

coord	Coordinates of the site(s), a matrix/dataframe with two columns (e.g. cbind(longitude, latitude)) (see examples).
level	a string or a vector of string, the length must match the number of rows of the parameter coord. This parameter gives the scale at which Feldpausch regions should be assigned. There are tree levels: • region: Models assign at sub-continent levels, value by default • continent: Models assign at the Africa, South America, Asia and Australia levels • world: Pantropical model

Value

The function returns a vector with the Feldpausch et al. (2012)'s regions that can be incorporated in the retrieveH function.

Author(s)

Arthur PERE

References

Feldpausch, T.R., et al. (2012). *Tree height integrated into pantropical forest biomass estimates.* Biogeosciences, 9, 3381–3403.

Examples

```
#' # One study site
lat <- 4.08
long <- -52.68
coord <- cbind(long, lat)

FeldRegion <- computeFeldRegion(coord)


# Several study sites (here three sites)
long <- c(-52.68, -51.12, -53.11)
lat <- c(4.08, 3.98, 4.12)
coord <- cbind(long, lat)

FeldRegion <- computeFeldRegion(coord)
```

`correctCoordGPS` *Correct the GPS coordinates*

Description

This function builds the most probable GPS coordinates of the plot corners from multiple GPS measurements.

Usage

```
correctCoordGPS(
  longlat = NULL,
  projCoord = NULL,
  coordRel,
  rangeX,
  rangeY,
  maxDist = 15,
  drawPlot = FALSE,
  rmOutliers = TRUE
)
```

Arguments

`longlat`	(optional) data frame with the coordinate in longitude latitude (eg. cbind(longitude, latitude)).
`projCoord`	(optional) data frame with the projected coordinate in X Y
`coordRel`	data frame with the relative coordinate in the same order than the longlat or projCoord
`rangeX`	a vector of length 2 giving the range for plot relative X coordinates
`rangeY`	a vector of length 2 giving the range for plot relative Y coordinates
`maxDist`	a numeric giving the maximum distance above which GPS measurements should be considered as outliers (by default 15 m)
`drawPlot`	a logical if you want to display a graphical representation
`rmOutliers`	a logical if you want to remove the outliers from coordinates calculation

Details

GPS coordinates should be either given in longitude latitude (longlat) or in projected coordinates (projCoord)

Value

If there are no outliers or rmOutliers = TRUE, a list with:

- `cornerCoords`: a data.frame with the coordinates of the corners

- `correctedCoord`: a data.frame with the adjusted coordinates given as input
- `polygon`: a spatial polygon
- `outliers`: index of coordinates lines considered as outliers, if any
- `codeUTM`: the UTM code of the coordinates if the parameter `longlat` is set

Author(s)

Arthur PERE, Maxime REJOU-MECHAIN

Examples

```
projCoord <- data.frame(
  X = c(
    runif(5, min = 9, max = 11), runif(5, min = 8, max = 12),
    runif(5, min = 80, max = 120), runif(5, min = 90, max = 110)
  ),
  Y = c(
    runif(5, min = 9, max = 11), runif(5, min = 80, max = 120),
    runif(5, min = 8, max = 12), runif(5, min = 90, max = 110)
  )
)
projCoord <- projCoord + 1000
coordRel <- data.frame(
  X = c(rep(0, 10), rep(100, 10)),
  Y = c(rep(c(rep(0, 5), rep(100, 5)), 2))
)

aa <- correctCoordGPS(
  projCoord = projCoord, coordRel = coordRel,
  rangeX = c(0, 100), rangeY = c(0, 100)
)
bb <- correctCoordGPS(
  projCoord = projCoord, coordRel = coordRel,
  rangeX = c(0, 100), rangeY = c(0, 100), rmOutliers = TRUE
)

correctCoordGPS(
  projCoord = projCoord, coordRel = coordRel,
  rangeX = c(0, 100), rangeY = c(0, 100), drawPlot = TRUE
)
```

---

correctTaxo *Correct trees taxonomy*

Description

This function corrects typos for a given taxonomic name using the Taxonomic Name Resolution Service (TNRS).

Usage

```
correctTaxo(
  genus,
  species = NULL,
  score = 0.5,
  useCache = FALSE,
  verbose = TRUE,
  accepted = FALSE
)
```

Arguments

genus	Vector of genera to be checked. Alternatively, the whole species name (genus + species) or (genus + species + author) may be given (see example).
species	(optional) Vector of species to be checked (same size as the genus vector).
score	Score of the matching (see https://tnrs.biendata.org/instructions) below which corrections are discarded.
useCache	logical. Whether or not use a cache to reduce online search of taxa names (NULL means use cache but clear it first)
verbose	logical. If TRUE various messages are displayed during process
accepted	logical. If TRUE accepted names will be returned instead of matched names. Cache will not be used as synonymy changes over time.

Details

This function create a file named correctTaxo.log (see Localisation), this file have the memory of all the previous requests, as to avoid the replication of time-consuming server requests.

By default, names are queried in batches of 500, with a 0.5s delay between each query. These values can be modified using options: `options(BIOMASS.batch_size=500)` for batch size (max 1000), `options(BIOMASS.wait_delay=0.5)` for delay (in seconds).

Value

The function returns a dataframe with the corrected (or not) genera and species.

Localisation

Cache path discovery protocol

1. BIOMASS.cache option set to an **existing** folder
2. **existing** user data folder `rappdirs::user_data_dir()`
   - On Linux : `~/.local/share/R/BIOMASS`
   - On Mac OS X : `~/Library/Application Support/R/BIOMASS`
   - On Windows 7 up to 10 : `C:\\Users\\<username>\\AppData\\Local\\R\\BIOMASS`
   - On Windows XP : `C:\\Documents and Settings\\<username>\\Data\\R\\BIOMASS`
3. fallback to R session tempdir

Author(s)

Ariane TANGUY, Arthur PERE, Maxime REJOU-MECHAIN, Guillaume CORNU

References

Boyle, B. et al. (2013). *The taxonomic name resolution service: An online tool for automated standardization of plant names*. BMC bioinformatics, 14, 1. doi:10.1186/1471-2105-14-16

Examples

```
## Not run:
correctTaxo(genus = "Astrocarium", species = "standleanum")
correctTaxo(genus = "Astrocarium standleanum")

## End(Not run)
```

`createCache` *Function used to create or activate a permanent cache.*

Description

Permanent cache is located by default in user data dir.

Usage

```
createCache(path = NULL)
```

Arguments

`path`	Use a custom path to host cache

Details

You can provide a custom path (that will be defined as a BIOMASS.cache option) but clearCache function will refuse to operate on it for security reasons.

Value

No return value, called for side effects

cutPlot *Divides one or more plots into subplots*

Description

This function divides a plot (or several plots) in subplots and returns the coordinates of the grid. These coordinates are calculated by a bilinear interpolation with the projected corner coordinates as references.

Usage

```
cutPlot(projCoord, plot, cornerNum, gridsize = 100, dimX = 200, dimY = 200)
```

Arguments

projCoord	A data frame containing the projected coordinates of plot corners, with X and Y on the first and second column respectively
plot	A vector indicating the plot codes
cornerNum	A vector with corners numbered from 1 to 4 for each plot, numbering must be in clockwise direction
gridsize	The size of the subplots
dimX	A vector indicating the size of the plot on the X axis, in meters and in the relative coordinates system (if a single value is supplied, it will be replicated for all plots)
dimY	A vector indicating the size of the plot on the Y axis, in meters and in the relative coordinates system (if a single value is supplied, it will be replicated for all plots)

Value

Returns a data-frame containing as many rows as there are corners corresponding to the subplots, and the following columns :

- plot: The plot code
- subplot: The automatically generated subplot code
- XRel: The relative coordinates on the X axis (defined by corners 1->4)
- YRel: The relative coordinates on the Y axis (defined by corners 1->2)
- XAbs: The absolute (projected) X coordinates
- YAbs: The absolute (projected) Y coordinates

Author(s)

Arthur PERE

Examples

```
coord <- data.frame(X = c(0, 200, 0, 200), Y = c(0, 0, 200, 200)) + 5000
cornerNum <- c(1, 2, 4, 3)
plot <- rep("plot1", 4)

cut <- cutPlot(coord, plot, cornerNum, gridsize = 100, dimX = 200, dimY = 200)

# plot the result
plot(coord, main = "example", xlim = c(4900, 5300), ylim = c(4900, 5300), asp = 1)
text(coord, labels = cornerNum, pos = 1)
points(cut$XAbs, cut$YAbs, pch = "+")
legend("bottomright", legend = c("orignal", "cut"), pch = c("o", "+"))
```

`divide_plot` *Divides one ore more plots into subplots*

Description

This function divides a plot (or several plots) into subplots in the relative coordinates system, and returns the coordinates of subplot corners.

Usage

```
divide_plot(
  corner_data,
  rel_coord,
  proj_coord = NULL,
  longlat = NULL,
  grid_size,
  grid_tol = 0.1,
  origin = NULL,
  tree_data = NULL,
  tree_coords = NULL,
  corner_plot_ID = NULL,
  tree_plot_ID = NULL,
  sd_coord = NULL,
  n = 100
)
```

Arguments

`corner_data`	A data frame, data frame extension, containing the plot corner coordinates. Typically, the output `$corner_coord` of the `check_plot_coord()` function.
`rel_coord`	A character vector of length 2, specifying the column names (resp. x, y) of the corner relative coordinates.

proj_coord	(optional, if longlat is not provided) A character vector of length 2, specifying the column names (resp. x, y) of the corner projected coordinates.
longlat	(optional, if proj_coord is not provided) A character vector of length 2, specifying the column names of the corner geographic coordinates (long,lat).
grid_size	A vector indicating the dimensions of grid cells (resp. X and Y dimensions). If only one value is given, grid cells will be considered as squares.
grid_tol	A numeric between (0;1) corresponding to the percentage of the plot area allowed to be excluded from the plot division (when grid_size doesn't match exactly plot dimensions).
origin	Alignment of the subplot grid, based on relative coordinates. If NULL (default), the grid is aligned to the origin corner of the relative coordinates. Alternatively provide a numeric vector of length 2, specifying the relative coordinates to which the grid should be aligned. This option is especially useful when grid_size doesn't match exactly plot dimensions.
tree_data	A data frame containing tree relative coordinates and other optional tree metrics (one row per tree).
tree_coords	A character vector of length 2, specifying the column names of the relative coordinates of the trees.
corner_plot_ID	If dealing with several plots: a vector indicating plot IDs for corners.
tree_plot_ID	If dealing with several plots: a vector indicating tree plot IDs.
sd_coord	used to propagate GPS measurements uncertainties to the subplot polygon areas and the ref_raster footprint in subplot_summary(). See Details.
n	used to propagate GPS measurements uncertainties: the number of iterations to be used (as in AGBmonteCarlo()). Cannot be smaller than 50 or larger than 1000.

Details

If corner coordinates in the projected coordinate system are provided (proj_coord), projected coordinates of subplot corners are calculated by a bilinear interpolation in relation with relative coordinates of plot corners. Be aware that this bilinear interpolation only works if the plot in the relative coordinates system is rectangular (ie, has 4 right angles).

In order to propagate GPS measurement uncertainties, the sd_coord argument has to be provided and must contains the average standard deviation of the GPS measurements for each corner on the X and Y axes (typically, the output $sd_coord of the check_plot_coord() function). If corner_data contains only one plot, sd_coord must be a numeric. If dealing with several plot, sd_coord must be a data frame of two columns named 'plot_ID' and 'sd_coord' containing respectively the plot IDs and the previous metric (again, see the output $sd_coord of the check_plot_coord() function).

Value

Returns a list containing:

- $sub_corner_coord: a data-frame containing as many rows as there are corners corresponding to the subplots, and the following columns :

  - plot_ID: If dealing with multiple plots: the plot code, else, a column containing an empty character
  - subplot_ID: The automatically generated subplot code, using the following rule : subplot_X_Y
  - x_rel and y_rel : the relative X-axis and Y-axis coordinates of subplots corners.
  - x_proj and y_proj : if proj_coord is provided, the projected X-axis and Y-axis coordinates of subplots corners

- $tree_data: the tree_data argument with the subplot_ID of each tree in the last column
- $UTM_code: if 'longlat' is provided, a data.frame containing the UTM code of the corner GPS coordinates for each plot
- $simu_coord: if sd_coord is provided, a list of n data-tables containing the simulated coordinates

Author(s)

Arthur PERE, Arthur BAILLY, John L. GODLEE

Examples

```
# One plot with repeated measurements of each corner
data("NouraguesPlot201")
check_plot201 <- check_plot_coord(
  corner_data = NouraguesPlot201,
  proj_coord = c("Xutm","Yutm"), rel_coord = c("Xfield","Yfield"),
  trust_GPS_corners = TRUE, draw_plot = FALSE)
subplots_201 <- divide_plot(
  corner_data = check_plot201$corner_coord,
  rel_coord = c("x_rel","y_rel"), proj_coord = c("x_proj","y_proj"),
  grid_size = 50)
subplots_201

# Assigning trees to subplots
data("NouraguesTrees")
plot201_trees <- NouraguesTrees[NouraguesTrees$Plot == 201,]
subplots_201 <- suppressWarnings(
  divide_plot(
    corner_data = check_plot201$corner_coord,
    rel_coord = c("x_rel","y_rel"), proj_coord = c("x_proj","y_proj"),
    grid_size = 50,
    tree_data = plot201_trees, tree_coords = c("Xfield","Yfield")))
head(subplots_201$sub_corner_coord)
head(subplots_201$tree_data)

# When grid dimensions (40m x 40m) don't fit perfectly plot dimensions
# an origin at (10 ; 10) will center the grid

  divide_plot(
    corner_data = check_plot201$corner_coord,
    rel_coord = c("x_rel","y_rel"),
    grid_size = c(40,40),
```

```
    grid_tol = 0.4,
    origin = c(10,10)
 )


# Dealing with multiple plots
data("NouraguesCoords")
nouragues_subplots <- suppressWarnings(
  divide_plot(
    corner_data = NouraguesCoords,
    rel_coord = c("Xfield","Yfield"), proj_coord = c("Xutm","Yutm"),
    corner_plot_ID = "Plot",
    grid_size = 50,
    tree_data = NouraguesTrees, tree_coords =  c("Xfield","Yfield"),
    tree_plot_ID = "Plot"))
head(nouragues_subplots$sub_corner_coord)
head(nouragues_subplots$tree_data)
```

`getTaxonomy` *Retrieve trees taxonomy*

Description

From given genus, the function finds the APG III family, and optionally the order, from the genus-Family database and the apgFamilies dataset

Usage

```
getTaxonomy(genus, findOrder = FALSE)
```

Arguments

`genus`	Vector of genus names
`findOrder`	(Boolean) If `TRUE`, the output will contain the taxonomical orders of the families.

Value

Data frame with the order (if `findOrder` is `TRUE`), family and genus.

Author(s)

Ariane TANGUY, Arthur PERE, Maxime REJOU-MECHAIN

Examples

```
# Find the Family of the Aphelandra genus
getTaxonomy("Aphelandra")
# ... and the order

getTaxonomy("Aphelandra", findOrder = TRUE)
```

`getWoodDensity` *Estimating wood density*

Description

The function estimates the wood density (WD) of the trees from their taxonomy or from their congeners using the global wood density database (Chave et al. 2009, Zanne et al. 2009) or any additional dataset. The WD can either be attributed to an individual at a species, genus, family or stand level.

Usage

```
getWoodDensity(
  genus,
  species,
  stand = NULL,
  family = NULL,
  region = "World",
  addWoodDensityData = NULL,
  verbose = TRUE
)
```

Arguments

`genus`	Vector of genus names
`species`	Vector of species names
`stand`	(optional) Vector with the corresponding stands of your data. If set, the missing wood densities at the genus level will be attributed at stand level. If not, the value attributed will be the mean of the whole tree dataset.
`family`	(optional) Vector of families. If set, the missing wood densities at the genus level will be attributed at family level if available.
`region`	Region (or vector of region) of interest of your sample. By default, Region is set to 'World', but you can restrict the WD estimates to a single region : • `AfricaExtraTrop`: Africa (extra tropical) • `AfricaTrop`: Africa (tropical) • `Australia`: Australia • `AustraliaTrop`: Australia (tropical)

- `CentralAmericaTrop`: Central America (tropical)
- `China`: China
- `Europe`: Europe
- `India`: India
- `Madagascar`: Madagascar
- `Mexico`: Mexico
- `NorthAmerica`: North America
- `Oceania`: Oceania
- `SouthEastAsia`: South-East Asia
- `SouthEastAsiaTrop`: South-East Asia (tropical)
- `SouthAmericaExtraTrop`: South America (extra tropical)
- `SouthAmericaTrop`: South America (tropical)
- `World`: World

`addWoodDensityData`
: A dataframe containing additional wood density data to be combined with the global wood density database. The dataframe should be organized in a dataframe with three (or four) columns: "genus","species","wd", the fourth column "family" is optional.

`verbose`
: A logical, give some statistic with the database

Details

The function assigns to each taxon a species- or genus- level average if at least one wood density value at the genus level is available for that taxon in the reference database. If not, the mean wood density of the family (if set) or of the stand (if set) is given.

The function also provides an estimate of the error associated with the wood density estimate (i.e. a standard deviation): a mean standard deviation value is given to the tree at the appropriate taxonomic level using the sd_10 dataset.

Value

Returns a dataframe containing the following information:

- `family`: (if set) Family
- `genus`: Genus
- `species`: Species
- `meanWD` (g/cm^3): Mean wood density
- `sdWD` (g/cm^3): Standard deviation of the wood density that can be used in error propagation (see sd_10 and `AGBmonteCarlo()`)
- `levelWD`: Level at which wood density has been calculated. Can be species, genus, family, dataset (mean of the entire dataset) or, if stand is set, the name of the stand (mean of the current stand)
- `nInd`: Number of individuals taken into account to compute the mean wood density

Author(s)

Maxime REJOU-MECHAIN, Arthur PERE, Ariane TANGUY

References

Chave, J., et al. *Towards a worldwide wood economics spectrum*. Ecology letters 12.4 (2009): 351-366. Zanne, A. E., et al. *Global wood density database*. Dryad. Identifier: http://hdl. handle. net/10255/dryad 235 (2009).

See Also

wdData, sd_10

Examples

```
# Load a data set
data(NouraguesTrees)

# Compute the Wood Density up to the genus level and give the mean wood density of the dataset

WD <- getWoodDensity(
  genus = NouraguesTrees$Genus,
  species = NouraguesTrees$Species
)


# Compute the Wood Density up to the genus level and then give the mean wood density per stand

WD <- getWoodDensity(
  genus = NouraguesTrees$Genus,
  species = NouraguesTrees$Species,
  stand = NouraguesTrees$plotId
)


# Compute the Wood Density up to the family level and then give the mean wood density per stand

WD <- getWoodDensity(
  family = NouraguesTrees$family,
  genus = NouraguesTrees$Genus,
  species = NouraguesTrees$Species,
  stand = NouraguesTrees$plotId
)
str(WD)
```

HDmethods *HDmethods*

Description

Methods used for modeling height-diameter relationship

Usage

```
loglogFunction(
  data,
  weight = NULL,
  method,
  bayesian,
  useCache,
  chains,
  thin,
  iter,
  warmup,
  ...
)

michaelisFunction(
  data,
  weight = NULL,
  bayesian,
  useCache,
  chains,
  thin,
  iter,
  warmup,
  ...
)

weibullFunction(
  data,
  weight = NULL,
  bayesian,
  useCache,
  chains,
  thin,
  iter,
  warmup,
  ...
)
```

Arguments

`data`	Dataset with the informations of height (H) and diameter (D)
`weight`	(optional) Vector indicating observation weights in the model.
`method`	In the case of the loglogFunction, the model is to be chosen between log1, log2 or log3.
`bayesian`	a logical. If FALSE (by default) the model is estimated using a frequentist framework (lm or nls). If TRUE, the model is estimated in a Bayesian framework using the brms package.
`useCache`	a logical. If bayesian = TRUE, determine wether to use the cache when building a Bayesian model (see Details).
`chains`	(only relevant if bayesian = TRUE): Number of Markov chains (defaults to 3), see `brms::brm()`
`thin`	(only relevant if bayesian = TRUE): Thinning rate, see `brms::brm()`
`iter`	(only relevant if bayesian = TRUE): number of total iterations per chain (including warmup; defaults to 5000), see `brms::brm()`
`warmup`	(only relevant if bayesian = TRUE): number of warmup (aka burnin) iterations (defaults to 1000), see `brms::brm()`
`...`	Further arguments passed to `brm()`, e.g: prior, cores, etc. See `brms::brm()`

Details

These functions model the relationship between tree height (H) and diameter (D). **loglogFunction** Compute two types of log model (log and log2) to predict H from D. The model can be:

- log 1: $log(H) = a + b * log(D)$ (equivalent to a power model)
- log 2: $log(H) = a + b * log(D) + c * log(D)^2$

michaelisFunction Construct a Michaelis Menten model of the form:

$$H = (A * D)/(B + D)$$

(A and B are the model parameters to be estimated)

weibullFunction Construct a three parameter Weibull model of the form:

$$H = a * (1 - exp(-(D/b)^c))$$

(a, b, c are the model parameters to be estimated)

Value

All the functions give an output similar to the one given by `stats::lm()`, obtained for `michaelisFunction` and `weibullFunction` from minpack.lm::nlsLM).

Result of a model (lm object if bayesian = FALSE, brm object if bayesian = TRUE)

Result of a model (nlsM object if bayesian = FALSE, brm object if bayesian = TRUE)

Result of a model (nlsM object if bayesian = FALSE, brm object if bayesian = TRUE)

Author(s)

Maxime REJOU-MECHAIN, Ariane TANGUY

References

Michaelis, L., & Menten, M. L. (1913). *Die kinetik der invertinwirkung*. Biochem. z, 49(333-369), 352. Weibull, W. (1951). *Wide applicability*. Journal of applied mechanics, 103. Baskerville, G. L. (1972). *Use of logarithmic regression in the estimation of plant biomass*. Canadian Journal of Forest Research, 2(1), 49-53.

See Also

`modelHD()`

`latlong2UTM` *Translate the long lat coordinate in UTM coordinate*

Description

Translate the long lat coordinate in UTM coordinate

Usage

```
latlong2UTM(coord)
```

Arguments

`coord`	Coordinates of the site(s), a matrix/dataframe with two columns (e.g. cbind(longitude, latitude)) (see examples).

Value

a data frame with :

- `long`: The longitude of the entry
- `lat`: The latitude of the entry
- `codeUTM`: The code `proj` for UTM
- `X`: The X UTM coordinate
- `Y`: The Y UTM coordinate

Examples

```
long <- c(-52.68, -51.12, -53.11)
lat <- c(4.08, 3.98, 4.12)
coord <- cbind(long, lat)

UTMcoord <- latlong2UTM(coord)
```

`modelHD` *Fitting height-diameter models*

Description

This function fits and compares (optional) height-diameter models.

Usage

```
modelHD(
  D,
  H,
  method = NULL,
  useWeight = FALSE,
  drawGraph = FALSE,
  plot = NULL,
  bayesian = FALSE,
  useCache = FALSE,
  chains = 3,
  thin = 5,
  iter = 5000,
  warmup = 500,
  ...
)
```

Arguments

`D`	Vector with diameter measurements (in cm). NA values are accepted but a minimum of 10 valid entries (i.e. having a corresponding height in H) is required.
`H`	Vector with total height measurements (in m). NA values are accepted but a minimum of 10 valid entries (i.e. having a corresponding diameter in D) is required.
`method`	Method used to fit the relationship. To be chosen between: • log1, log2 – log 1: $(log(H) = a + b * log(D))$ (equivalent to a power model) – log 2: $(log(H) = a + b * log(D) + c * log(D)^2)$ • weibull: $H = a * (1 - exp(-(D/b)^c))$ • michaelis: $H = (A * D)/(B + D)$ If `NULL`, all the methods will be compared.
`useWeight`	If weight is `TRUE`, model weights will be $(D^2) * H$ (i.e. weights are proportional to tree volume, so that larger trees have a stronger influence during the construction of the model).
`drawGraph`	If `TRUE`, a graphic will illustrate the relationship between H and D. Only if argument `plot` is null.

plot	(optional) a vector of character containing the plot ID's of the trees (linked to D and H). Must be either one value, or a vector of the same length as D. This argument is used to build stand-specific HD models.
bayesian	a logical. If FALSE (by default) the model is estimated using a frequentist framework (lm or nls). If TRUE, the model is estimated in a Bayesian framework using the brms package.
useCache	a logical. If bayesian = TRUE, determine wether to use the cache when building a Bayesian model (see Details).
chains	(only relevant if bayesian = TRUE): Number of Markov chains (defaults to 3), see brms::brm()
thin	(only relevant if bayesian = TRUE): Thinning rate, see brms::brm()
iter	(only relevant if bayesian = TRUE): number of total iterations per chain (including warmup; defaults to 5000), see brms::brm()
warmup	(only relevant if bayesian = TRUE): number of warmup (aka burnin) iterations (defaults to 1000), see brms::brm()
...	Further arguments passed to brm(), e.g: prior, cores, etc. See brms::brm()

Details

All the back transformations for log-log models are done using the Baskerville correction ($0.5 * RSE^2$, where RSE is the Residual Standard Error).

If useCache = TRUE and this is the first time the model is being built, the model will be saved as a .rds file in the defined cache path (see createCache()). If useCache = TRUE and the model has already been built using the user cache, the model will be loaded and updated to avoid wasting time re-compiling it. If useCache = NULL, the cache is first cleared before building the model.

Value

If plot is NULL or has a single value, a single list is returned. If there is more than one plot, multiple embedded lists are returned with plots as the list names.

If model is not null (model comparison), returns a list :

- input: list of the data used to construct the model (list(H, D))
- model: outputs of the model (same outputs as given by stats::lm(), stats::nls())
- residuals: Residuals of the model
- method: Name of the method used to construct the model
- predicted: Predicted height values
- RSE: Residual Standard Error of the model
- RSElog: Residual Standard Error of the log model (NULL if other model)
- fitPlot: a ggplot object containing the model fitting plot
- weighted: a logical indicating whether weights were used during the fit

If the parameter model is null, the function return a plot with all the methods for comparison, the function also returns a data.frame with:

- `method`: The method that had been used to construct the plot
- `RSE`: Residual Standard Error of the model
- `RSElog`: Residual Standard Error of the log model (`NULL` if other model)
- `Average_bias`: The average bias for the model

Author(s)

Maxime REJOU-MECHAIN, Arthur PERE, Ariane TANGUY, Arthur Bailly

See Also

`retrieveH()`

Examples

```
# Load a data set
data(NouraguesHD)

# Fit H-D models for the Nouragues dataset
 HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H, drawGraph = TRUE)

# For a selected model
HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H,
                   method = "log2", drawGraph = TRUE)

# Using weights
HDmodel <- modelHD(
  D = NouraguesHD$D, H = NouraguesHD$H,
  method = "log2", useWeight = TRUE,
  drawGraph = TRUE)

# With multiple stands (plots)
HDmodel <- modelHD(
  D = NouraguesHD$D, H = NouraguesHD$H,
  method = "log2", useWeight = TRUE,
  plot = NouraguesHD$plotId, drawGraph = TRUE)

### Using log2 bayesian model
## Not run: HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H,
  method = "log2", bayesian = TRUE, useCache = TRUE)
plot(HDmodel$model)
## End(Not run)

### Using weibull bayesian model (time consuming)
# As the algorithm is likely to find numerous local minima,
# defining priors is strongly recommended (see "Some tricks" part in the vignette)
# Also, since  model parameters and chain iterations are strongly correlated,
# an increase of 'thin', 'iter' and 'warmup' may be required.
## Not run: HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H,
 method = "weibull", bayesian = TRUE, useCache = TRUE,
 thin = 20, iter = 12000, warmup = 2000,
```

```
 prior = c(brms::set_prior(prior = "uniform(0,80)",
                           lb = 0, ub = 80, class = "b", nlpar = "a"),
           brms::set_prior(prior = "uniform(0,100)",
                           lb = 0, ub = 100, class = "b", nlpar = "b"),
           brms::set_prior(prior = "uniform(0.1,0.9)",
                           lb = 0.1, ub = 0.9, class = "b", nlpar = "c")))
## End(Not run)
```

`NouraguesCoords` *Nouragues plot coordinates*

Description

Dataset containing the corner coordinates of 4 plots of 'Petit Plateau' in Nouragues forest (French Guiana).

Usage

```
data(NouraguesCoords)
```

Format

A data frame with 16 observations (GPS measurements) of the 8 following variables :

- `Site`: Name of the site set up in the Nouragues forest
- `Plot`: Plot ID of the site
- `Xfield`: Corner location on the x-axis in the local coordinate system (defined by the 4 corners of the plot)
- `Yfield`: Corner location on the y-axis in the local coordinate system
- `Xutm`: Corner location on the x-axis in the UTM coordinate system
- `Yutm`: Corner location on the y-axis in the UTM coordinate system
- `Long`: Corner longitude coordinate
- `Lat`: Corner latitude coordinate

References

Jaouen, Gaëlle, 2023, "Nouragues forest permanent plots details", doi:10.18167/DVN1/HXKS4E, CIRAD Dataverse, V2

Examples

```
data(NouraguesCoords)
str(NouraguesCoords)
```

NouraguesHD *Height-Diameter data*

Description

Dataset from two 1-ha plots from the Nouragues forest (French Guiana)

Usage

```
data("NouraguesHD")
```

Format

A data frame with 1051 observations on the following variables :

- `plotId`: Names of the plots
- `genus`: Genus
- `species`: Species
- `D`: Diameter (cm)
- `H`: Height (m)
- `lat`: Latitude
- `long`: Longitude

References

Réjou-Méchain, M. et al. (2015). *Using repeated small-footprint LiDAR acquisitions to infer spatial and temporal variations of a high-biomass Neotropical forest* Remote Sensing of Environment, 169, 93-101.

Examples

```
data(NouraguesHD)
str(NouraguesHD)
```

NouraguesPlot201 *Nouragues plot 201 coordinates*

Description

Simulated corner coordinates of Nouragues 'Petit plateau' plot 201. The original coordinates have been modified to make the plot non-squared, and 10 repeated measurements of each corner have been simulated adding a random error to x and y coordinates.

Usage

```
data(NouraguesPlot201)
```

Format

A data frame with 40 (simulated GPS measurements) of the 8 following variables :

- `Site`: Name of the site set up in the Nouragues forest
- `Plot`: Plot ID of the site
- `Xfield`: Corner location on the x-axis in the local coordinate system (defined by the 4 corners of the plot)
- `Yfield`: Corner location on the y-axis in the local coordinate system
- `Xutm`: Corner location on the x-axis in the UTM coordinate system
- `Yutm`: Corner location on the y-axis in the UTM coordinate system
- `Long`: Corner longitude coordinate
- `Lat`: Corner latitude coordinate

References

Jaouen, Gaëlle, 2023, "Nouragues forest permanent plots details", doi:10.18167/DVN1/HXKS4E, CIRAD Dataverse, V2

Examples

```
data(NouraguesPlot201)
str(NouraguesPlot201)
```

NouraguesTrees *Nouragues forest dataset*

Description

This dataset contains 4 of the 12 plots of 'Petit Plateau' permanent plots fifth census, 2012, Nouragues forestTree dataset (French Guiana). For educational purposes, some virtual trees have been added in the dataset. Dead trees have been removed.

Usage

```
data(NouraguesTrees)
```

Format

A data frame with 2050 observations (trees) of the 8 following variables :

- `site`: Name of the site set up in the Nouragues forest
- `plot`: Plot ID
- `Xfield`: Tree location on the x-axis in the local coordinate system (defined by the 4 corners of the plot)
- `Yfield`: Tree location on the y-axis in the local coordinate system
- `family`: Tree family
- `genus`: Tree genus
- `species`: Tree species
- `D`: Tree diameter (in cm)

References

'Petit Plateau' permanent plots fifth census, 2012, Nouragues forest, https://doi.org/10.18167/DVN1/TZ1RL9, CIRAD Dataverse, V1

Examples

```
data(NouraguesTrees)
str(NouraguesTrees)
```

---

numberCorner *Get the UTM coordinates with the corner of the plot*

Description

Get the UTM coordinates from the latitude and longitude of the corners of a plot. The function also assign a number to the corners in a clockwise or counterclockwise way, with the number 1 for the XY origin. Corner numbering is done as followed:

- axis X: the corner 1 to the corner 2
- axis Y: the corner 1 to the corner 4

Usage

```
numberCorner(longlat = NULL, projCoord = NULL, plot, origin, clockWise)
```

Arguments

`longlat`	(optional) data frame with the coordinates in longitude latitude (eg. cbind(longitude, latitude)).
`projCoord`	(optional) data frame with the projected coordinates in X Y
`plot`	A vector of codes (names) of the plots
`origin`	A logical vector with TRUE corresponding of the origin of the axis of each plot.
`clockWise`	A logical, whether the numbering should be done in a clockwise (TRUE) or counterclockwise (FALSE) way.

Value

A data frame with:

- `plot`: The code of the plot
- `X`: The coordinates X in UTM
- `Y`: The coordinates Y in UTM
- `corner`: The corner numbers

Author(s)

Arthur PERE, Maxime REJOU-MECHAIN

Examples

```
coord <- data.frame(X = c(0, 200, 0, 200), Y = c(0, 0, 200, 200)) + 5000
plot <- rep("plot1", 4)
origin <- c(FALSE, FALSE, TRUE, FALSE)

# if you turn clock wise
corner <- numberCorner(projCoord = coord, plot = plot, origin = origin, clockWise = TRUE)

# Plot the plot
plot(coord, asp = 1)
text(coord, labels = corner$corner, pos = 1)


# Using a counterclockwise way
corner <- numberCorner(projCoord = coord, plot = plot, origin = origin, clockWise = FALSE)

# Plot the plot
plot(coord, asp = 1)
text(coord, labels = corner$corner, pos = 1)
```

---

predictHeight *Tree height predictions*

Description

The function predicts height from diameter based on a fitted model. As the predict() function for brms models takes ~10 minutes to run, predictions are calculated using the coefficients from the models directly.

Usage

```
predictHeight(D, model, err = FALSE, plot = NULL)
```

Arguments

D	a n x m matrix containing tree diameters (in cm), where n is the number of trees and m is the number of Monte Carlo simulations (m = 1 if no error propagation).
model	The output of the modelHD() function.
err	If TRUE, An error is taken randomly from a normal distribution with a mean of zero and a standard deviation equaled to the residual standard error of the model (RSE). Only used for the Monte Carlo approach (see AGBmonteCarlo()), otherwise it should be let as FALSE, the default case.
plot	(optional) Plot ID, must be either one value, or a vector of the same length as D. This argument is used to build stand-specific HD models.

Details

In the case where the error is FALSE and the model is a log-log model, we use the Baskerville correction, a bias correction factor used to get unbiased backtransformation values.

Value

Returns a vector of total tree height (in m).

Author(s)

Arthur BAILLY

See Also

minpack.lm::nlsLM()

---

procrust *Procrust analysis*

Description

Do a procrust analysis. X is the target matrix, Y is the matrix we want to fit to the target. This function returns a translation vector and a rotation matrix After the procrust problem you **must** do the rotation before the translation. **Warning : The order of the value on both matrix is important**

Usage

```
procrust(X, Y)
```

Arguments

X	the target matrix
Y	the matrix we want to fit to the target

Value

A list with the translation vector and the matrix of rotation

Author(s)

Arthur PERE

`retrieveH` *Retrieving tree height from models*

Description

From the diameter and either i) a model, ii) the coordinates of the plot or iii) the region, this function gives an estimate of the total tree height.

Usage

```
retrieveH(D, model = NULL, coord = NULL, region = NULL, plot = NULL)
```

Arguments

`D`	Vector of diameters.
`model`	A model output by the function `modelHD()`.
`coord`	Coordinates of the site(s), either a vector (e.g. c(longitude, latitude)) or a matrix/dataframe with two columns (e.g. cbind(longitude, latitude)).
`region`	Area of your dataset to estimate tree height thanks to Weibull-H region-, continent-specific and pantropical models proposed by Feldpausch et al. (2012). To be chosen between: • `Africa`: Africa • `CAfrica`: Central Africa • `EAfrica`: Eastern Africa • `WAfrica`: Western Africa • `SAmerica`: Southern America • `BrazilianShield`: Brazilian Shield • `ECAmazonia`: East-Central Amazonia • `GuianaShield`: Guiana Shield • `WAmazonia`: Western Amazonia • `SEAsia`: South-Eastern Asia • `NAustralia`: Northern Australia • `Pantropical`: Pantropical
`plot`	(optional) Plot ID, must be either one value, or a vector of the same length as D. This argument is used to build stand-specific HD models.

Value

Returns a list with:

- H: Height predicted by the model
- RSE Residual Standard Error of the model, or a vector of those for each plot

Author(s)

Ariane TANGUY, Maxime REJOU-MECHAIN, Arthur PERE

References

Feldpausch et al. *Tree height integrated into pantropical forest biomass estimates*. Biogeosciences (2012): 3381-3403.

Chave et al. *Improved allometric models to estimate the aboveground biomass of tropical trees*. Global change biology 20.10 (2014): 3177-3190.

See Also

modelHD()

Examples

```
# Load a database
data(NouraguesHD)
model <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H, method = "log2")

# If any height model is available
H <- retrieveH(D = NouraguesHD$D, model = model)

# If the only data available are the coordinates of your spot
n <- length(NouraguesHD$D)
coord <- cbind(long = rep(-52.68, n), lat = rep(4.08, n))

H <- retrieveH(D = NouraguesHD$D, coord = coord)


# If the only data available is the region of your spot
H <- retrieveH(D = NouraguesHD$D, region = "GuianaShield")
```

---

subplot_summary *Summarise and display tree information by subplot*

Description

After applying the divide_plot() function, this function summarises with any defined function the desired tree metric (including AGB simulations calculated by the AGBmonteCarlo() function) by sub-plot and displays the plot representation.

Usage

```
subplot_summary(
  subplots,
  value = NULL,
  AGB_simu = NULL,
  draw_plot = TRUE,
  per_ha = TRUE,
  fun = sum,
  ref_raster = NULL,
  raster_fun = mean,
  ...
)
```

Arguments

subplots	output of the divide_plot() function
value	a character indicating the column in subplots$tree_data to be summarised (or character vector to summarise several metrics at once)
AGB_simu	a n x m matrix containing individual AGB where n is the number of tree and m is the number of monte carlo simulation. Typically, the output '$AGB_simu' of the AGBmonteCarlo() function.
draw_plot	a logical indicating whether the plot design should be displayed
per_ha	a logical indicating whether the metric summary should be per hectare (or, if summarising several metrics at once: a logical vector corresponding to each metric (see examples))
fun	the function to be applied on tree metric of each subplot (or, if summarising several metrics at once: a list of functions named according to each metric (see examples))
ref_raster	A SpatRaster object from terra package, typically a chm raster created from LiDAR data. Note that in the case of a multiple attributes raster, only the first variable "z" will be summarised.
raster_fun	the function (or a list of functions) to be applied on raster values of each subplot.
...	optional arguments to fun

Value

a list containing the following elements:

- tree_summary: a summary of the metric(s) per subplot
- polygon: a simple feature collection of the summarised subplot's polygon
- plot_design : a ggplot object (or a list of ggplot objects) that can easily be modified

If 'AGB_simu' is provided, the function also return $long_AGB_simu: a data.table containing the resulting AGBD, the extracted raster values (if ref_raster is provided) and the coordinates of the center per subplot and per simulation.

Author(s)

Arthur Bailly

Examples

```
# One plot with repeated measurements of each corner
data("NouraguesPlot201")
data("NouraguesTrees")

check_plot201 <- check_plot_coord(
  corner_data = NouraguesPlot201,
  proj_coord = c("Xutm","Yutm"), rel_coord = c("Xfield","Yfield"),
  trust_GPS_corners = TRUE, draw_plot = FALSE)
subplots_201 <- suppressWarnings(
  divide_plot(
    corner_data = check_plot201$corner_coord,
    rel_coord = c("x_rel","y_rel"), proj_coord = c("x_proj","y_proj"),
    grid_size = 50,
    tree_data =  NouraguesTrees[NouraguesTrees$Plot == 201,],
    tree_coords = c("Xfield","Yfield")))
# Sum summary (by default) of diameter
subplots_201_sum <- subplot_summary(subplots_201 , value = "D", draw_plot = FALSE)
subplots_201_sum$tree_summary

  subplots_201_sum$plot_design

# 9th quantile summary (for example) of diameter
subplots_201_quant <- subplot_summary(subplots_201 , value = "D", draw_plot = FALSE,
                                      fun = quantile, probs=0.9)


# Dealing with multiple plots and metrics
## Not run:
  data("NouraguesCoords")
  nouragues_subplots <- suppressWarnings(
  divide_plot(
     corner_data = NouraguesCoords,
    rel_coord = c("Xfield","Yfield"), proj_coord = c("Xutm","Yutm"),
    corner_plot_ID = "Plot",
    grid_size = 50,
    tree_data = NouraguesTrees, tree_coords =  c("Xfield","Yfield"),
    tree_plot_ID = "Plot"))
  nouragues_mult <- subplot_summary(nouragues_subplots ,
                                    value = c("D","D","x_rel"),
                                    fun = list(D=sum,D=mean,x_rel=mean),
                                    per_ha = c(T,F,F),
                                    draw_plot = FALSE)
  nouragues_mult$tree_summary
  nouragues_mult$plot_design$`201`[[1]]
  nouragues_mult$plot_design$`201`[[2]]
  nouragues_mult$plot_design$`201`[[3]]
```

```
## End(Not run)

# Dealing with AGB simulations, coordinates uncertainties of corners and a CHM raster
## Not run:
  NouraguesTrees201 <- NouraguesTrees[NouraguesTrees$Plot == 201,]
  nouragues_raster <- terra::rast(
    system.file("extdata", "NouraguesRaster.tif",
                package = "BIOMASS", mustWork = TRUE)
  )

  # Modelling height-diameter relationship
  HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H, method = "log2")
  # Retrieving wood density values
  Nouragues201WD <- getWoodDensity(
    genus = NouraguesTrees201$Genus,
    species = NouraguesTrees201$Species)
  # MCMC AGB simulations
  resultMC <- AGBmonteCarlo(
    D = NouraguesTrees201$D, Dpropag = "chave2004",
    WD = Nouragues201WD$meanWD, errWD = Nouragues201WD$sdWD,
    HDmodel = HDmodel,
    n = 200
  )
  # Dividing plot 201 with coordinates uncertainties
  nouragues_subplots <- suppressWarnings(
    divide_plot(
      corner_data = check_plot201$corner_coord,
      rel_coord = c("x_rel","y_rel"), proj_coord = c("x_proj","y_proj"),
      grid_size = 50,
      tree_data = NouraguesTrees201, tree_coords =  c("Xfield","Yfield"),
      sd_coord = check_plot201$sd_coord, n = 200
    )
  )
  # Summary (may take few minutes to extract all raster metrics)
  res_summary <- subplot_summary(
    subplots = nouragues_subplots,
    AGB_simu = resultMC$AGB_simu,
    ref_raster = nouragues_raster, raster_fun = mean)

  res_summary$tree_summary
  res_summary$plot_design[[1]]
  head(res_summary$long_AGB_simu)

## End(Not run)
```

---

summaryByPlot *Summarise by plot the posterior distribution of AGB values*

Description

This function summarises the matrix AGB_val given by the function AGBmonteCarlo() by plot.

Usage

```
summaryByPlot(AGB_val, plot, drawPlot = FALSE)
```

Arguments

AGB_val	Either the matrix resulting from the AGBmonteCarlo() function (AGB_simu element of the list), or simply the output of the AGBmonteCarlo() function itself.
plot	Vector corresponding to the plots code (plots ID)
drawPlot	A logic indicating whether the graphic should be displayed or not

Details

If some trees belong to an unknown plot (i.e. NA value in the plot arguments), their AGB values are randomly assigned to a plot at each iteration of the AGB monte Carlo approach.

Value

a data frame where:

- plot: the code of the plot
- AGB: AGB value at the plot level
- Cred_2.5: the 2.5\
- Cred_97.5: the 97.5\

Examples

```
# Load a database
data(NouraguesHD)
data(NouraguesTrees)

# Modelling height-diameter relationship
HDmodel <- modelHD(D = NouraguesHD$D, H = NouraguesHD$H, method = "log2")

# Retrieving wood density values

  NouraguesWD <- getWoodDensity(NouraguesTrees$Genus, NouraguesTrees$Species,
                                stand = NouraguesTrees$plotId)


# Propagating errors

  resultMC <- AGBmonteCarlo(
    D = NouraguesTrees$D, WD = NouraguesWD$meanWD,
    errWD = NouraguesWD$sdWD, HDmodel = HDmodel )

  # The summary by plot
  summaryByPlot(AGB_val = resultMC$AGB_simu, plot = NouraguesTrees$Plot)
```

Index