Package ‘lidaRtRee’

April 11, 2021

Type Package
Version 3.0.3
Title Forest Analysis with Airborne Laser Scanning (LiDAR) Data
Date 2021-04-01
Author Jean-Matthieu Monnet [aut, cre]
Maintainer Jean-Matthieu Monnet <jean-matthieu.monnet@inrae.fr>
Description Provides functions for forest analysis using airborne laser scanning (LiDAR remote sensing) data: tree detection (method 1 in Eysn et al. (2015) <doi:10.3390/f6051721>) and segmentation; forest parameters estimation and mapping with the area-based approach. It includes complementary steps for forest mapping: co-registration of field plots with LiDAR data (Monnet and Mermin (2014) <doi:10.3390/f5092307>); extraction of both physical (gaps, edges, trees) and statistical features from LiDAR data useful for e.g. habitat suitability modeling (Glad et al. (2020) <doi:10.1002/rse2.117>); model calibration with ground reference, and maps export.
URL https://gitlab.irstea.fr/jean-matthieu.monnet/lidaRtRee
BugReports https://gitlab.irstea.fr/jean-matthieu.monnet/lidaRtRee/-/issues
Depends R (>= 3.5.0)
Imports graphics, stats, methods, grDevices, sp, raster, imager, leaps, gvlma, car, reldist, lidR (>= 3.1.0)
License GPL-3
LazyData TRUE
RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation no
Repository CRAN
Date/Publication 2021-04-10 23:10:02 UTC
R topics documented:

ABAinference  ..................................................  3
ABAmodel  .........................................................  4
ABAmodelCombineStrata  .........................................  5
ABAmodelMetrics  ................................................  7
ABAmodelPlot  ......................................................  8
ABApredict  ........................................................  9
AddVegetationIndices  ............................................ 10
BoxcoxTr  .......................................................... 10
chmchablais3  .................................................... 11
cimg2Raster  ...................................................... 12
circle2Raster  ..................................................... 13
cleanRaster  ........................................................ 14
cloudMetrics  ....................................................... 15
cloudTreeMetrics  .................................................. 16
coregistration  ..................................................... 17
createDisk  .......................................................... 19
demFiltering  ....................................................... 20
detectEdge  .......................................................... 21
detectEllipses4Crown  ............................................ 22
detectGap  ............................................................ 23
detectHeightRegression  .......................................... 25
detectHist  ............................................................ 26
detectHistStack  .................................................... 27
detectiBoxcoxTr  ................................................... 28
detectiBoxcoxTrBiasCor  ......................................... 29
detectlaschablais3  ............................................... 30
detectlmaCheck  ................................................... 31
detectmaximaDetection  ........................................... 32
detectmaximaSelection  ............................................ 33
detectplot2Dmatched  ............................................. 34
detectplotTreeInventory  .......................................... 35
detectpointList2SPDF  ............................................ 36
detectpoints2DSM  .................................................. 37
detectpoints2DTM  .................................................. 38
detectpoints2terrainStats  ....................................... 39
detectpolar2Projected  ............................................ 40
detectquatremontagnes  .......................................... 41
detectraster2Cimg  ................................................ 42
detectrasterChullMask  ........................................... 43
detectrasterLocalmax  ............................................ 44
detectrasterMetrics  ............................................... 45
detectrasters2Cor  ................................................ 46
detectrastersMovingCor  ......................................... 48
detectrasterXYMask  .............................................. 49
detectrasterZonalStats  .......................................... 50
detectsegAdjust  ................................................... 51
ABAinference computes inference from area-based model and predicted values

**Description**

computes inference from area-based model and predicted values

**Usage**

```r
ABAinference(
  modell,  
  r.predictions,  
  type = c("SRS", "ED", "D", "STR", "SYNT"),  
  r.mask = NULL  
)
```

**Arguments**

- `modell`: a model returned by `ABAmodel` or `ABAmodelCombineStrata`
- `r.predictions`: raster of predicted values
- `type`: string vector specifying which estimators should be computed (one or several in "SRS", "ED", "D", "STR", "SYNT")
- `r.mask`: raster to mask region of interest (NA values), may contain post-stratification categories (should be integer, positive values)

**Value**

a dataframe with estimation of parameter value and standard deviation of estimation for all required estimators.
ABA model

Calibrates and validates area-based models

Description

The function can first apply a Box-Cox transformation to the dependent variable, in order to normalize its distribution, or a log transformation to the whole dataset. Then it uses `regsubsets` to find the 20 linear regressions with the best adjusted-R2 among combinations of at most nmax independent variables. Each model can then be tested regarding the following linear model assumptions are checked:

- tests performed by `gvlma`
- the variance inflation factor is below 5 (models with two or more independent variables)
- no partial p.value of variables in the model is below 0.05

The model with the highest adjusted-R2 among those fulfilling the required conditions is selected. A leave-one-out cross validation (LOO CV) is performed by fitting the model coefficients using all observations except one and applying the resulting model to predict the value for the remaining observation. In case a transformation was performed beforehand, a bias correction is applied. LOO CV statistics are then computed.

Usage

```r
ABAmodel(
  variable, predictors,
  transform = "none",
  nmax = 3,
  test = c("partial_p", "vif", "gvlma"),
  xy = NULL,
  threshold = NULL
)
```

Arguments

- **variable**: vector. dependent variable values
- **predictors**: data.frame. independent variables (columns: metrics, lines: observations). Row names are used for the output predicted values
- **transform**: string. transformation to be applied to data ("none", "boxcox": Box-Cox transformation applied only to the dependent variable, "log": log transformation applied to both dependent and independent variables)
- **nmax**: numeric. maximum number of independent variables in the model
- **test**: vector. which tests should be satisfied by the models, one to three in "partial_p", "vif", "gvlma"
xy data.frame or matrix of easting and northing coordinates of observations: not used in the function but exported in the result for use in further inference functions

threshold vector of length two. minimum and maximum values of threshold to apply to predicted values

Value

a list with three elements

- model: list with one regression model (output from \texttt{lm}),
- stats: model statistics (root mean square error estimated in leave-one-out cross validation, coefficient of variation of rmse, p-value of wilcoxon test of observed and predicted values, p-value of t-test of observed and predicted values, p-value of anova of observed and predicted values, correlation of observed and predicted values, R2 of observed and predicted values, variance of regression residuals)
- values: data.frame with observed and values predicted in cross-validation.

See Also

\texttt{ABAmodelCombineStrata} for combining models calibrated on different strata, \texttt{ABAmodelPlot} for plotting model cross-validation results, \texttt{regsubsets} for variable selection, \texttt{lmaCheck} for linear model assumptions check, \texttt{iBoxcoxTrBiasCor} for reverse Box-Cox transformation with bias correction.

Examples

```r
# load Quatre Montagnes dataset
data(quatremontagnes)
# build ABA model for basal area, with all metrics as predictors
model.aba <- ABAmodel(quatremontagnes$G.m2.ha, quatremontagnes[, 9:76],
                      transform = "boxcox", nmax = 3)
# summary of regression model
summary(model.aba$model)
# validation statistics
model.aba$stats
# observed and predicted values
summary(model.aba$values)

# plot field values VS predictions in cross-validation
ABAmodelPlot(model.aba, main = "Basal area")
```

\textit{ABAmodelCombineStrata} \textit{Combines a list of ABA models into a single ABA model object}
Description

Combines a list of models (obtained with \texttt{ABAmodel}) into a single object. Typically used to merge stratum-specific models into one object. Validation statistics are computed for the combined strata, making it easier to compare prediction performance with an unstratified model.

Usage

\texttt{ABAmodelCombineStrata(model.list, plotsId = NULL)}

Arguments

- \texttt{model.list}: list. stratum-specific models returned by \texttt{ABAmodel}
- \texttt{plotsId}: vector. "plotsId" for ordering row names in the "values" element of the output list

Value

a list with three elements

- \texttt{model}: a list of regression models corresponding to each stratum (output from \texttt{lm}).
- \texttt{stats}: model statistics of each stratum-specific model (as in \texttt{ABAmodel}) plus one line corresponding to statistics for all strata (COMBINED)
- \texttt{values}: data.frame with observed and values predicted in cross-validation, and information on which stratum it belongs to.

See Also

\texttt{ABAmodel} for calibrated ABA model, \texttt{ABAmodelPlot} for plotting model cross-validation results.

Examples

```r
# load Quatre Montagnes dataset
data(quatremontagnes)
# initialize list of models
model.ABA.stratified <- list()
# calibrate basal area prediction model for each stratum
for (i in levels(quatremontagnes$stratum)) {
    subsample <- which(quatremontagnes$stratum==i)
    model.ABA.stratified[[i]] <-
    ABAmodel(quatremontagnes[subsample, "G.m2.ha"],
              quatremontagnes[subsample, 9:76], transform="boxcox", nmax=4,
              xy = quatremontagnes[subsample,c("X", "Y")])
}
# combine models in single object
model.ABA.stratified <- ABAmodelCombineStrata(model.ABA.stratified,
                                            quatremontagnes$plotId)
# display content of output list
model.ABA.stratified$model
model.ABA.stratified$stats
```
ABAmodelMetrics

summary(model.ABA.stratified$values)

# plot field values VS predictions in cross-validation
ABAmodelPlot(model.ABA.stratified)

ABAmodelMetrics  Function for area-based metrics computation

Description

Predefined function usable in cloud_metrics or cloudMetrics. Applies a minimum height threshold to the point cloud and computes the following metrics:

1. for all points: total number \( n_{tot} \), percentage of points above minimum height \( p_{h_{min}} \), percentage of points in height bins \( H_{propZ1-Z2} \),
2. for first return points: percentage above minimum height \( p_{1st.h_{min}} \),
3. for all points above minimum height: height metrics returned by stdmetrics_z and intensity metrics returned by stdmetrics_i
4. for first returns above minimum height: \( mCH \) and \( sdCH \) as proposed by Bouvier et al.

Usage

ABAmodelMetrics(z, i, rn, c, hmin = 2, breaksH = NULL)

Arguments

\( z, i, rn, c \)  Height, Intensity, ReturnNumber and Classification

\( hmin \)  numeric, height threshold for low points removal before metrics computation

\( breaksH \)  vector, breaks for height histogram proportion computation

Format

An object of class formula of length 2.

References


See Also

cloud_metrics, stdmetrics, cloudMetrics
Examples

data(laschablais3)

# extract four point clouds from LAS object
llas <- list()
llas[["A"]]<- lidR::clip_circle(laschablais3, 974350, 6581680, 10)
llas[["B"]]<- lidR::clip_circle(laschablais3, 974390, 6581680, 10)
llas[["C"]]<- lidR::clip_circle(laschablais3, 974350, 6581640, 10)
# normalize point clouds
llas <- lapply(llas, function(x) {lidR::normalize_height(x, lidR::tin())})

# compute metrics
cloudMetrics(llas, -ABAmodelMetrics(  
  Z, Intensity, ReturnNumber, Classification, 2, c(-Inf, 0, 2, 10, 20, +Inf)))

ABAmodelPlot

Plots observed VS values predicted in leave one out cross validation of an ABAmodel

Description

Plots observed VS values predicted in leave one out cross validation of an ABAmodel

Usage

ABAmodelPlot(modell, disp.text = F, col = NULL, ...)

Arguments

modell list. as returned by ABAmodel
disp.text boolean. indicates if points should be labeled with id
col color to be passed to plot, default is black for single models, depends on stratum in stratified models
...
other parameters to be passed to plot. xlab and ylab are automatically setup

Value

nothing

Examples

# load Quatre Montagnes dataset
data(quatremontagnes)
# build ABA model for basal area, with all metrics as predictors
model.aba <- ABAmodel(quatremontagnes$G.m2.ha, quatremontagnes[, 9:76],  
  transform = "boxcox", nmax = 3)

# plot field values VS predictions in cross-validation
ABAmodelPlot(model.aba, main = "Basal area")
Mapping of ABA prediction models

Description

Applies calibrated area-based prediction models output of `ABAmodel` to a RasterStack of metrics to obtain a raster of predictions

Usage

```r
ABApredict(aba.model, metrics.map, stratum = NULL, addError = FALSE)
```

Arguments

- `aba.model`: model returned by `ABAmodel` or `ABAmodelCombineStrata`
- `metrics.map`: RasterStack. metrics returned e.g by `grid_metrics`
- `stratum`: string. indicates which layer of metrics.map contains the stratum in model
- `addError`: boolean. indicates whether errors sampled from a normal distribution $N(0, \sigma(\text{residuals}))$ should be added to fitted values; implemented only for log transformation case

Value

A raster of predictions obtained by applying the model `aba.model` to the observations in `metrics.map`

See Also

`ABAmodel` for model fitting and `ABAmodelCombineStrata` for combining stratified models, `cleanRaster` for applying spatial mask and value thresholds to a raster.

Examples

```r
# load data
data(quatremontagnes)

# build model
ABA.model <- ABAmodel(quatremontagnes$G.m2.ha, quatremontagnes[,9:76],
  transform = "boxcox")

# build example raster to apply model
quatremontagnes$X <- rep(1:8, 12)
quatremontagnes$Y <- rep(1:12, each = 8)
metrics.map <- raster::rasterFromXYZ(quatremontagnes[, c(2,3,9:76)])
predict.map <- lidaRtRee::ABApredict(ABA.model, metrics.map)

# plot map
raster::plot(predict.map, main = "predictions")
```
AddVegetationIndices  Add vegetation indices on a IRC image

Description
Computes vegetations indices from the Red, Green and Infra-Red bands of an IRC image and adds them as additional bands or columns. Indices are listed on https://www.l3harrisgeospatial.com/docs/broadbandgreenness.html

Usage
AddVegetationIndices(r, all = FALSE)

Arguments
- r: raster stack or data.frame. Should contain bands or columns with names nir, r, g
- all: boolean. indicates whether all indices should be computed; default:FALSE, only grvi, sr and ndvi are calculated

Value
a RasterStack or data.frame with added bands or columns

Examples
df <- data.frame(nir=c(110, 150, 20), r=c(25, 50, 30), g=c(10, 60, 10))
AddVegetationIndices(df, all=TRUE)

BoxcoxTr  Box-Cox Transformation

Description
Box-Cox Transformation

Usage
BoxcoxTr(x, lambda)

Arguments
- x: vector or RasterLayer. values to be transformed
- lambda: numeric. parameter of Box-Cox transformation
Value

a vector or RasterLayer of transformed values

See Also

iBoxcoxTr inverse Box-Cox transformation, iBoxcoxTrBiasCor inverse Box-Cox transformation with bias correction.

Examples

```r
x <- 1:10
BoxcoxTr(x, -2)
BoxcoxTr(x, 0)
BoxcoxTr(x, 0.5)
BoxcoxTr(x, 2)

# plot functions
curve(BoxcoxTr(x, 1.5), 1, 5, main = "Box Cox transform", xlab = "x",
ylab = "Boxcox(x, lambda)", col = "red")
curve(BoxcoxTr(x, -2), 1, 5, col = "green", add = TRUE)
curve(BoxcoxTr(x, 0), 1, 5, col = "blue", add = TRUE)
curve(BoxcoxTr(x, 0.5), 1, 5, col = "black", add = TRUE)
curve(BoxcoxTr(x, 1), 1, 5, col = "pink", add = TRUE)
legend("topleft", legend = rev(c(-2, 0, 0.5, 1, 1.5, "lambda")),
col = rev(c("green", "blue", "black", "pink", "red", NA)), lty = 1)
```

chmchablais3  

Canopy height model (Chablais 3 plot)

Description

Canopy height model computed from airborne laser scanning data acquired in July 2010.

Usage

data(chmchablais3)

Format

A raster object

References

Examples

```
data(chmchablais3)
chmchablais3
raster::plot(chmchablais3)
```

---

### cimg2Raster

**Cimg to RasterLayer conversion**

**Description**

converts a cimg object to a RasterLayer object

**Usage**

```
cimg2Raster(cimg, rasterLayer = NULL)
```

**Arguments**

- `cimg` raster object. raster of canopy height model, preferably filtered to avoid effect of holes on volume and surface computation
- `rasterLayer` raster object. defines the extent and projection of conversion result

**Value**

A RasterLayer

**See Also**

`raster2Cimg`

**Examples**

```
data(chmchablais3)

# convert rasterLayer to cimg object
chm.cim <- raster2Cimg(chmchablais3)

# apply filtering
chm.cim.filt <- demFiltering(chm.cim,
    nlFilter = "Closing",
    nlSize = 3,
    sigmap = 0)$non.linear.image

# convert to RasterLayer
chm.filt <- cimg2Raster(chm.cim.filt, chmchablais3)

# plot rasterLayer
raster::plot(chmchablais3)
```
circle2Raster

# plot cimg object
plot(chm.cim)

# plot filtered cimg object
plot(chm.cim.filt)

# plot filtered rasterLayer
raster::plot(chm.filt)

---

circle2Raster  

**Description**

Creates an empty raster which extents corresponds to the circle specified by center coordinates, radius and optional buffer size.

**Usage**

```r
circle2Raster(X, Y, radius, resolution = 0.5, buffer = 0.5)
```

**Arguments**

- `X`: numeric. easting coordinate of plot center in meters
- `Y`: numeric. northing coordinate of plot center in meters
- `radius`: numeric. plot radius in meters
- `resolution`: numeric. raster resolution in meters
- `buffer`: numeric. buffer to be added to plot radius in meters

**Value**

A raster object

**Examples**

```r
circle2Raster(100, 100, 20, 1, 5)
```
cleanRaster | Applies thresholds and mask to a rasterLayer object

Description

Applies a lower and upper thresholds to the values of the input raster. If the mask input is provided, first all NA values in the raster are set to 0, then the raster in multiplied by the mask. Cells to be masked should therefore have a NA value in the mask raster object.

Usage

cleanRaster(rast, minmax = c(-Inf, +Inf), mask = NULL)

Arguments

- rast: raster object.
- minmax: vector of two numeric values. minimum and maximum thresholds to apply to 'rast' values
- mask: raster object. mask to be applied (multiplication with input raster 'rast')

Value

- a raster object

Examples

# load data
data(quatremontagnes)
# build model
ABA.model <- lidaRtRee::ABAmodel(quatremontagnes$G.m2.ha, quatremontagnes[, 9:76], transform = "boxcox")
# build example raster to apply model
quatremontagnes$X <- rep(1:8, 12)
quatremontagnes$Y <- rep(1:12, each = 8)
metrics.map <- raster::rasterFromXYZ(quatremontagnes[, c(2,3,9:76)])
predict.map <- lidaRtRee::ABApredict(ABA.model, metrics.map)
# create raster mask
mask <- predict.map
# set values to 1 or NA
raster::values(mask) <- rep(c(1, 1, NA), each = 32)
# apply thresholds and mask
predict.map.clean <- cleanRaster(predict.map, c(40, 70), mask)

# plot maps
raster::plot(predict.map, main = "Predictions")
raster::plot(mask, main = "Mask", legend = FALSE)
raster::plot(predict.map.clean, main = "Cleaned predictions")
cloudMetrics  Computes metrics on list of point clouds

Description

Computes metrics for a list of LAS objects (should be normalized point clouds). Calls the function cloud_metrics on each element and then arranges the results in a data.frame.

Usage

cloudMetrics(
  llasn,
  func = ~lidR::stdmetrics(X, Y, Z, Intensity, ReturnNumber, Classification, dz = 1)
)

Arguments

llasn  list of LAS objects
func   function. function applied on each element to compute metrics, default function is stdmetrics from package lidR

Value

A data frame with metrics in columns corresponding to LAS objects of the list (lines)

See Also

cloud_metrics, stdmetrics, ABAmodelMetrics

Examples

data(laschablais3)

# extract four point clouds from LAS object
llas <- list()
llas[["A"]] <- lidR::clip_circle(laschablais3, 974350, 6581680, 10)
llas[["B"]] <- lidR::clip_circle(laschablais3, 974390, 6581680, 10)
llas[["C"]] <- lidR::clip_circle(laschablais3, 974350, 6581640, 10)
# normalize point clouds
llas <- lapply(llas, function(x) {lidR::normalize_height(x, lidR::tin())})

# compute metrics
cloudMetrics(llas)

# compute metrics with user-defined function
# mean and standard deviation of first return points above 10 m
user.func <- function(z, rn, hmin = 10)
{
  # first return above hmin subset
cloudTreeMetrics

Computes metrics on trees detected in list of point clouds.

dummy <- which(z >= hmin & rn == 1)
return(list(
  mean.z = mean(z[dummy]),
  sd.z = stats::sd(z[z>hmin])
))
cloudMetrics(11as, func=user.func(Z, ReturnNumber, 10))

description

Extracts summary statistics on trees for each LAS object in a list:

Usage

cloudTreeMetrics(llasn, XY, plot.radius, res = 0.5, func, ...)

Arguments

llasn         list of LAS objects
XY            a dataframe or matrix with XY coordinates of plot centers
plot.radius   numeric. plot radius in meters
res           numeric. resolution of canopy height model computed with points2DSM before
tree segmentation
func          a function to be applied to the attributes of extracted trees (return from internal
call to treeExtraction function) to compute plot level metrics
...           other parameters to be passed to treeSegmentation

Details

• calls treeSegmentation to segment trees and then treeExtraction to extract their features
• computes 'TreeCanopy.coverInPlot' (proportion of surface of disk of interest which is covered
  by segmented trees), 'TreeCanopy.meanHeightInPlot' (mean canopy height inside intersection
  of tree segments and disk of interest)
• removes detected trees located outside of the disk of interest defined by their centers and radius
• computes summary statistics of extracted tree features based on a user-defined function (de-
  fault is stdTreeMetrics)

Value

a dataframe with tree metrics in columns corresponding to LAS objects of the list (lines)

See Also

treeSegmentation, treeExtraction, stdTreeMetrics
Examples

data(laschablais3)

# extract three point clouds of 10 m radius from LAS object
llas <- list()
llas[[1]] <- lidR::clip_circle(laschablais3, 974350, 6581680, 10)
llas[[2]] <- lidR::clip_circle(laschablais3, 974390, 6581680, 10)
llas[[3]] <- lidR::clip_circle(laschablais3, 974350, 6581640, 10)
# normalize point clouds
llas <- lapply(llas, function(x) {lidR::normalize_height(x, lidR::tin())})

# compute tree metrics restricted to disks of radius 8 m.
cloudTreeMetrics(llas,
               cbind(c(974350, 974390, 974350), c(6581680, 6581680, 6581640)),
               8, res=0.5)

# compute metrics with user-defined function
# number of detected trees between 20 and 30 meters and their mean height
user.func <- function(x)
  {
    dummy <- x$h[which(x$h>20 & x$h<30)]
    data.frame(Tree.between.20.30=length(dummy), Tree.meanH=mean(dummy))
  }
cloudTreeMetrics(llas,
               cbind(c(974350, 974390, 974350), c(6581680, 6581680, 6581640)),
               8, res=0.5, func=user.func)

Description

Computes the correlation between the canopy height model and a virtual canopy height model
simulated from tree locations, for different translations of tree inventory positions, and outputs the
translation corresponding to best estimated co-registration.

Usage

coregistration(chm, trees, mask = NULL, buffer = 19, step = 0.5, dm = 2, plot = TRUE)
Arguments

- **chm**: raster. canopy height model
- **trees**: data.frame. the first two columns contain xy coordinates, and the third is the value to correlate to the chm (e.g. tree heights or diameters)
- **mask**: raster. raster mask of tree inventory area
- **buffer**: numeric. radius of the circular buffer area of possible translations
- **step**: numeric. increment step of translations within buffer area to compute correlation values, should be a multiple of raster resolution
- **dm**: numeric. minimum distance between two local maxima in meters
- **plot**: boolean. whether to display the results or not

Value

A list with two elements: first the correlation raster returned by `rastersMovingCor`, second a `data.frame` returned by `rasterLocalmax`

References


See Also

`rastersMovingCor, rasterLocalmax`

Examples

```r
# tree inventory
trees <- data.frame(x=c(22.2, 18.3, 18.1), y =c(22.1, 22.7, 18.4), z=c(15,10,15))

# mask of inventory area
# empty raster with extent
tree.mask <- circle2Raster(20, 20, 9, resolution=1)
# fill binary mask
tree.mask <- rasterXYMask(rbind(c(20, 20), c(20, 20)), c(9,9), tree.mask, binary=TRUE)

# simulate chm raster
chm <- raster::raster()
raster::extent(chm) <- c(0,40,0,40)
raster::res(chm) <- 1
xy <- raster::xyFromCell(chm,1:length(chm))

# add Gaussian surfaces to simulate tree crowns
z1 <- trees$z[1]*exp(-((xy[,1]-trees$x[1])^2+(xy[,2]-trees$y[1])^2/2)*trees$z[1]/50)
z2 <- trees$z[2]*exp(-((xy[,1]-trees$x[2])^2+(xy[,2]-trees$y[2])^2/2)*trees$z[2]/50)
z3 <- trees$z[3]*exp(-((xy[,1]-trees$x[3])^2+(xy[,2]-trees$y[3])^2/2)*trees$z[3]/50)
chm <- raster::rasterFromXYZ(cbind(xy,pmax(z1,z2,z3)))+rnorm(length(z1),0,1))
```
# translate trees
trees$x <- trees$x + 1
trees$y <- trees$y + 2

coreg <- coregistration(chm, trees, mask=tree.mask, buffer=5, step=1, dm=1, plot=FALSE)
coreg$local.max[, c("dx1", "dy1")]

# plot raster
raster::plot(coreg$correlation.raster)
abline(h=0, lty=2)
abline(v=0, lty=2)
# add location of two local maxima
graphics::points(coreg$local.max[1, c("dx1", "dx2")],
coreg$local.max[1, c("dy1", "dy2")], cex=c(1, 0.5), pch=3, col="red")

createDisk

Disk-shaped matrix mask

Description

Creates a matrix with TRUE values shaping a centered disk

Usage

createDisk(width = 5)

Arguments

width numeric. disk width in pixels, should be an uneven number

Value

A matrix with 1 for pixels inside the disk, 0 outside

Examples

createDisk(7)
demFiltering

Image pre-processing (non-linear filtering and Gaussian smoothing)

Description

applies two filters to an image:

1. A non-linear filter: closing (mclosing) with disk kernel, or median (medianblur) with square kernel
2. A 2D Gaussian smoother (The deriche filter is applied on both dimensions). Value-dependent smoothing is possible

Usage

demFiltering(
  dem,
  nlFilter = "Closing",
  nlSize = 5,
  sigmap = 0.3,
  padding = TRUE
)

Arguments

dem cimg object (e.g. obtained with as.cimg) or Raster Layer object (e.g. obtained with raster)
nlFilter string. type of non-linear filter to apply: "None", "Closing" or "Median"
nlSize numeric. kernel width in pixel for non-linear filtering
sigmap numeric or matrix. if a single number is provided, sigmap is the standard deviation of the Gaussian filter in pixel, 0 corresponds to no smoothing. In case of matrix, the first column corresponds to the standard deviation of the filter, and the second to thresholds for image values (e.g. a filter of standard deviation specified in line i is applied to pixels in image which values are between thresholds indicated in lines i and i+1). Threshold values should be ordered in increasing order.
padding boolean. Whether image should be padded by duplicating edge values before filtering to avoid border effects

Value

A list of two cimg or a RasterStack objects: image after non-linear filter and image after both filters

See Also

maximaDetection, filters of imager package: mclosing, medianblur, deriche
Examples

data(chmchablais3)

# filtering with median and Gaussian smoothing
im <- demFiltering(chmchablais3, nlFilter="Median", nlSize=3, sigmap=0.8)

# filtering with median filter and value-dependent Gaussian smoothing
# (less smoothing for values between 0 and 15)
im2 <- demFiltering(chmchablais3, nlFilter="Median", nlSize=3,
                   sigmap=cbind(c(0.2,0.8), c(0,15)))

# plot original image
raster::plot(chmchablais3, main="Initial image")

# plot image after median filter
raster::plot(im$non.linear.image, main="Median filter")

# plot image after median and Gaussian filters
raster::plot(im$smoothed.image, main="Smoothed image")

# plot image after median and value-dependent Gaussian filters
raster::plot(im2$smoothed.image, main="Value-dependent smoothing")

edgeDetection

Edge detection in gap image

Description

Performs edge detection on a gap image (e.g. output from function gapDetection). The gap image is compared to a gap image which has undergone a dilation or erosion to identify edges of gaps.

Usage

detecting(gaps, inside = TRUE)

Arguments

gaps  raster object. gaps image where 1 represents gaps and 0 non-gaps areas
inside boolean. defines where the edge is extracted: either inside the gaps (an erosion is applied to the gaps image) or outside (a dilation is applied)

Value

A raster object where edges are labelled as 1.

See Also

gapDetection
Examples

data(chmchablais3)

# fill NA values in canopy height model
chmchablais3[is.na(chmchablais3)] <- 0

# gap detection with distance larger than canopy height / 2
gaps <- gapDetection(chmchablais3, ratio=2, gap.max.height=1, min.gap.surface=10,
gapReconstruct=TRUE)

# edge detection
edges.inside <- edgeDetection(!is.na(gaps$gap.id))
edges.outside <- edgeDetection(!is.na(gaps$gap.id), inside=FALSE)

# edge proportion
sum(raster::values(edges.inside))/(nrow(edges.inside)*ncol(edges.inside))
sum(raster::values(edges.outside))/(nrow(edges.outside)*ncol(edges.outside))

# plot original image
raster::plot(chmchablais3, main="Initial image")

# plot binary image of gaps
raster::plot(gaps$gap.id>0, main="Gaps", legend=FALSE)

# plot edges
raster::plot(edges.inside, main="Edges (inside)", legend=FALSE)
raster::plot(edges.outside, main="Edges (outside)", legend=FALSE)

---

ellipses4Crown

Create elliptical polygons from centres and extensions in four directions

description

creates polygons from the union of four quarters of ellipses, specified by the ellipse center, and
maximum extension in two directions

Usage

ellipses4Crown(x, y, n, s, e, w, id = NULL, step = pi/12, angle.offset = 0)

Arguments

x, y vectors of numerics. Coordinates of ellipses centers
n, s, e, w vectors of numerics. Coordinates of ellipses extension in the north, south, east
and west directions
id vector of strings. id of each polygon
step numeric. Angular step for the modelling of ellipses
angle.offset numeric. Angle offset to tilt ellipses, positive values rotates clockwise
gapDetection

Value

a list of data.frame containing the coordinates of polygons

See Also

pointList2SPDF

Examples

# compute coordinates of ellipses
ellipses1 <- ellipses4Crown(c(0,10), c(0,10), c(2,2), c(3,4), c(2.5,3), c(2,3),
   id=c("A", "B"))

ellipses1[["A"]]

# tilted ellipse
ellipses2 <- ellipses4Crown(c(0,10), c(0,10), c(2,2), c(3,4), c(2.5,3), c(2,3),
   angle.offset=pi/6)

ellipses2[[2]]

# draw ellipses in black, tilted ellipses in red
plot(ellipses1[[1]], type="l", asp=1, xlim=c(-5,15), ylim=c(-5,15))
lines(ellipses1[[2]])
lines(ellipses2[[1]], col="red")
lines(ellipses2[[2]], col="red")

gapDetection

Gap detection in a Canopy Height Model

Description

Performs gaps detection in a canopy height model. Function demFiltering is first applied to the canopy height model to remove artefacts. Gaps are then extracted based on several criteria:

1. Vegetation height must be smaller than a threshold
2. Gap width must be large enough, depending on surrounding canopy height; distance to surrounding vegetation is tested with morphological closings
3. Gap must have a minimum surface

Usage

gapDetection(
   chm,
   ratio = 2,
   gap.max.height = 1,
   min.gap.surface = 25,
   max.gap.surface = +Inf,
   closing.height.bin = 1,
   nlFilter = "Median",
   nlsize = 3,
   gapReconstruct = FALSE
)
gapDetection

Arguments

chm
raster object. canopy height model

ratio
numeric. maximum ratio between surrounding canopy height and gap distance
(a pixel belongs to the gap only if for any vegetation pixel around it, the distance
to the vegetation pixel is larger than pixel height/ratio). If ratio is set to NULL,
this criterion is not taken into account

gap.max.height
numeric. maximum canopy height to be considered as gap

min.gap.surface
numeric. minimum gap surface

max.gap.surface
numeric. maximum gap surface

closing.height.bin
numeric. height bin width for morphological closing of gaps to test ratio be-
tween canopy height and gap distance

nlFilter
string. type of non-linear filter to apply to canopy height model to remove arte-
facts, should be an option of demFiltering

nlsize
numeric. kernel width in pixel for non-linear filtering

gapReconstruct
boolean. default behaviour is that areas that do not fulfill the ratio criterion are
removed from gaps. If set to TRUE, in case some pixels of a gap fulfill the
distance criterion, the connected pixels that fulfill the height criterion are also
integrated to it.

Value

A list of three raster objects: raster with gap labels, raster with gap surface, canopy height model
after filter.

See Also
demFiltering, edgeDetection

Examples

data(chmchablais3)

# fill NA values in canopy height model
chmchablais3[is.na(chmchablais3)] <- 0

# gap detection with distance larger than canopy height / 2
gaps <- gapDetection(chmchablais3, ratio=2, gap.max.height=1, min.gap.surface=0)

# gap detection with distance larger than canopy height / 2
# and reconstruction of border areas
gaps1 <- gapDetection(chmchablais3, ratio=2, gap.max.height=1, min.gap.surface=0,
gapReconstruct=TRUE)

# gap detection without distance criterion
gaps2 <- gapDetection(chmchablais3, ratio=NULL, gap.max.height=1, min.gap.surface=0)
# gap id and corresponding surface for third detection parameters
table(raster::values(gaps2$gap.id))*raster::res(gaps2$gap.id)[1]^2

# plot original image
raster::plot(chmchablais3, main="Initial image")

# plot binary image of gaps
raster::plot(gaps$gap.id>0, main="Gaps", legend=FALSE)
raster::plot(gaps1$gap.id>0, main="Gaps, with reconstruction", legend=FALSE)
raster::plot(gaps2$gap.id>0, main="Gaps, no width criterion", legend=FALSE)

# plot filtered CHM
raster::plot(gaps2$filled.chm, main="Filtered CHM")

--------

heightRegression Regression of detected heights VS reference heights

Description

Computes a linear regression model between the reference heights and the detected heights of matched pairs.

Usage

heightRegression(lr, ld, matched, plot = TRUE, species = NULL, ...)

Arguments

- lr: data.frame or matrix. 3D coordinates (X Y Height) of reference positions
- ld: data.frame or matrix. 3D coordinates (X Y Height) of detected positions
- matched: data.frame. contains pair indices, typically returned by `treeMatching`
- plot: boolean. indicates whether results should be plotted
- species: vector of strings. species for standardized color use by call to `speciesColor`
- ...: arguments to be passed to methods, as in `plot`

Value

A list with two elements. First one is the linear regression model, second one is a list with stats (root mean square error, bias and standard deviation of detected heights compared to reference heights).

See Also

treeMatching
Examples

```r
# create tree locations and heights
ref.trees <- cbind(c(1,4,3,4,2), c(1,1,2,3,4), c(15,18,20,10,11))
def.trees <- cbind(c(2,2,4,4), c(1,3,4,1), c(16,19,9,15))

# tree matching
match1 <- treeMatching(ref.trees, def.trees)

# height regression
reg <- heightRegression(ref.trees, def.trees, match1,
    species = c("ABAL", "ABAL", "FASY", "FASY", "ABAL"),
    asp = 1, xlim = c(0, 21), ylim = c(0, 21))
summary(reg$lm)
reg$stats
```

**histDetection**

*Histogram of detection*

**Description**

Displays the histogram of tree heights of three categories: true detections, omissions, and false detections.

**Usage**

```r
histDetection(lr, ld, matched, plot = TRUE)
```

**Arguments**

- `lr` data.frame or matrix. 3D coordinates (X Y Height) of reference positions
- `ld` data.frame or matrix. 3D coordinates (X Y Height) of detected positions
- `matched` data.frame. contains pair indices, typically returned by `treeMatching`
- `plot` boolean. should the histogram be displayed or not

**Value**

A list with three numerics: numbers of true detections, omissions and false detections

**See Also**

`treeMatching`
Examples

# create reference and detected trees
ref.trees <- cbind(c(1,4,3,4,2), c(1,1,2,3,4), c(15,18,20,10,11))
def.trees <- cbind(c(2,2,4,4), c(1,3,4,1), c(16,19,9,15))
#
# tree matching with different buffer size
match1 <- treeMatching(ref.trees, def.trees)
match2 <- treeMatching(ref.trees, def.trees, deltaGround=2, hPrec=0)
#
# corresponding number of detections
histDetection(ref.trees, def.trees, match1)
histDetection(ref.trees, def.trees, match2)

histStack

Description

Stacked histogram

Usage

histStack(x, breaks, col = NULL, breaksFun = paste, ...)

Arguments

x
  list of vectors. values for each category
breaks
  vector. breaks for histogram bins
col
  vector. colors for each category
breaksFun
  function for breaks display
...
  arguments to be passed to methods, as in plot

Value

no return
iBoxcoxTr

Inverse Box-Cox transformation

Description
Inverse Box-Cox transformation

Usage
iBoxcoxTr(x, lambda)

Arguments
x vector or RasterLayer. values to be transformed
lambda numeric. parameter of Box-Cox transformation

Value
a vector or RasterLayer of transformed values

See Also
BoxcoxTr Box-Cox transformation, iBoxcoxTrBiasCor inverse Box-Cox transformation with bias correction.

Examples
x <- 1:10
iBoxcoxTr(x, 0)
iBoxcoxTr(x, 0.5)
iBoxcoxTr(x, 2)
iBoxcoxTr(BoxcoxTr(x, 2), 2)

# plot functions
curve(iBoxcoxTr(x, 0), 0, 3, col = "blue", main = "inverse Box Cox transf.", xlab = "x", ylab = "inverse Boxcox(x, lambda)"
curve(iBoxcoxTr(x, 1.5), 0, 3, col = "red", add = TRUE)
curve(iBoxcoxTr(x, 0.5), 0, 3, col = "black", add = TRUE)
curve(iBoxcoxTr(x, 1), 0, 3, col = "pink", add = TRUE)
legend("topleft", legend = c("lambda", 0, 0.5, 1, 1.5),
col = c(NA, "blue", "black", "pink", "red"), lty = 1)
**Description**

Inverse Box-Cox transform with bias correction as suggested by Pu & Tiefelsdorf (2015). Here 'varmod' is not the local prediction variance as suggested in the paper but the model residuals variance. For variance computation, use 'n-p' instead of 'n-1', with 'p' the number of variables in the model.

**Usage**

```r
iBoxcoxTrBiasCor(x, lambda, varmod)
```

**Arguments**

- `x`: vector or RasterLayer. values to be transformed
- `lambda`: numeric. parameter of Box-Cox transformation
- `varmod`: numeric. model residuals variance

**Value**

a vector or RasterLayer

**References**

Xiaojun Pu and Michael Tiefelsdorf, 2015. A variance-stabilizing transformation to mitigate biased variogram estimation in heterogeneous surfaces with clustered samples. doi: 10.1007/9783319-227863_24

**See Also**

`BoxcoxTr` Box-Cox transformation, `iBoxcoxTr` inverse Box-Cox transformation.

**Examples**

```r
x <- 1:10
iBoxcoxTr(x, 0.3)
iBoxcoxTrBiasCor(x, 0.3, 0)
iBoxcoxTrBiasCor(x, 0.3, 2)

# plot functions
curve(iBoxcoxTr(x, 0.3), 0, 3, col = "blue",
     main = "inverse Box Cox transf., lambda = 0.3",
     xlab = "x", ylab = "inverse Boxcox(x, lambda = 0.3)"
)
curve(iBoxcoxTrBiasCor(x, 0.3, 1), 0, 3, col = "red", add = TRUE)
curve(iBoxcoxTrBiasCor(x, 0.3, 2), 0, 3, col = "black", add = TRUE)
legend("topleft", legend = c("residuals variance = 2", "bias corrected variance = 2",),
     col = c("blue", "red", "black"))
```
Description

Airborne laser scanning data over the Chablais 3 plot, acquired in 2009 by Sintegra, copyright INRAE

Usage

data(laschablais3)

Format

An object of class \texttt{LAS}

Details

Additional information about the data

- Sensor: RIEGL LMS-Q560)
- EPSG code of coordinates system: 2154

Source

Monnet J.-M. INRAE

References


Examples

data(laschablais3)
laschablais3
# display point cloud
lidR::plot(laschablais3)
lmaCheck

Checks linear model assumptions of a multiple regression model

Description

The performed tests are:

- partial p.values calculated by \texttt{lm} are all below a given value
- tests implemented by \texttt{gvlma}
- variance inflation factors calculated by \texttt{vif} are all below a given value

Usage

\texttt{lmaCheck(formule, df, max.pvalue = 0.05, max.vif = 5)}

Arguments

- \texttt{formule}: formula. model to be evaluated
- \texttt{df}: data.frame. data to evaluate the model
- \texttt{max.pvalue}: numeric. maximum p-value of variables included in the model
- \texttt{max.vif}: numeric. maximum variance inflation factor of variables included in the model

Value

a one line data.frame with 5 columns.

- a string: evaluated formula
- a numeric: the adjusted R squared of the model
- a boolean: do all variables in the model have a partial p-value < \texttt{max.pvalue}
- a boolean: are all tests implemented by \texttt{gvlma} false
- a boolean: is the variance inflation factor computed with \texttt{vif} of all variables < \texttt{max.vif}

Examples

# load Quatre Montagnes dataset
data(quatremontagnes)
# fit lm model
model <- \texttt{lm(G.m2.ha ~ zmax + zq95, data = quatremontagnes)}
lmaCheck(eval(model$call[[2]]), quatremontagnes)
# trying with Box-Cox transformation of dependent variable
# and other independent variables
model <- \texttt{lm(BoxcoxTr(G.m2.ha, -0.14) ~ Tree.meanH + Tree.density + zpcum7, data = quatremontagnes)}
lmaCheck(eval(model$call[[2]]), quatremontagnes)
maximaDetection

Local maxima extraction on image

Description

Variable window size maxima detection is performed on the image to extract local maxima position and calculate the window size where they are global maxima. Gaussian white noise is added to the image to avoid adjacent maxima due to neighbor pixels with identical value.

Usage

maximaDetection(dem, dem.res = 1, max.width = 21, jitter = TRUE)

Arguments

dem
  cimg object (e.g. as created by cimg) or RasterLayer object (e.g. obtained with raster)
dem.res
  numeric. image resolution, in case dem is a rasterLayer object, dem.res is extracted from the object by res
max.width
  numeric. maximum kernel width in pixel to check for local maximum
jitter
  boolean. indicates if noise should be added to image values to avoid the adjacent maxima due to the adjacent pixels with equal values

Value

A cimg object or RasterLayer object which values are the radius (n) in meter of the square window (width 2n+1) where the center pixel is global maximum

See Also

demFiltering, maximaSelection

Examples

data(chmchablais3)

# maxima detection
maxi <- maximaDetection(chmchablais3)

# plot original image
raster::plot(chmchablais3, main="Initial image")

# plot maxima image
raster::plot(maxi, main="Local maxima")
**maximaSelection**

Image maxima selection based on values and neighborhood of local maxima

**Description**

In a maxima image (output of `maximaDetection`), sets values to zero for pixels which

1. value in the initial image (from which maxima were detected) are below a threshold
2. values in the maxima image (corresponding to the radius of the neighborhood where they are global maxima) are below a threshold depending on the initial image value.

**Usage**

`maximaSelection(maxi, dem.nl, hmin = 0, dmin = 0, dprop = 0)`

**Arguments**

- `maxi`: cimg object or RasterLayer object. image with local maxima (typically output from `maximaDetection`, image values correspond to neighborhood radius on which pixels are global maxima in the initial image)
- `dem.nl`: cimg object. initial image from which maxima were detected
- `hmin`: numeric. minimum value in initial image for a maximum to be selected
- `dmin`: numeric. intercept term for selection of maxima depending on neighborhood radius: `maxi >= dmin + dem.nl * dprop`
- `dprop`: numeric. proportional term for selection of maxima depending on neighborhood radius: `maxi >= dmin + dem.nl * dprop`

**Value**

A cimg object or rasterLayer object which values are the radius (n) in meter of the square window (width 2n+1) where the center pixel is global maximum and which fulfill the selection criteria

**See Also**

`maximaDetection`

**Examples**

```r
data(chmchablais3)

# maxima detection
maxi <- maximaDetection(chmchablais3)

# several maxima selection settings
selected.maxi.hmin <- maximaSelection(maxi, chmchablais3, hmin=15)
selected.maxi.dm <- maximaSelection(maxi, chmchablais3, dmin=2.5)
```
```r
selected.maxi <- maximaSelection(maxi, chmchablais3, dm=1, dprop=0.1)

# corresponding count number of remaining maxima
table(raster::values(maxi))
table(raster::values(selected.maxi.hmin))
table(raster::values(selected.maxi.dm))
table(raster::values(selected.maxi))

# plot original image
raster::plot(chmchablais3, main="Initial image")

# plot maxima images, original and first case
raster::plot(maxi, main="Local maxima")
raster::plot(selected.maxi, main="Selected maxima")
```

---

**plot2Dmatched**  
*Plot of matched pairs of detected and reference trees*

### Description

Plot of matched pairs of detected and reference trees

### Usage

```r
plot2Dmatched(lr, ld, matched, chm = NULL, plotBoundary = NULL, ...)
```

### Arguments

- **lr**  
data.frame or matrix. 3D coordinates (X Y Height) of reference positions
- **ld**  
data.frame or matrix. 3D coordinates (X Y Height) of detected positions
- **matched**  
data.frame. contains pair indices, typically returned by `treeMatching`
- **chm**  
raster object. raster for background display
- **plotBoundary**  
Spatialpolygon. plot boundaries for display
- **...**  
Additional arguments to be used by `plot`

### Value

no return

### See Also

`treeMatching`, `histDetection`
Examples

```r
# create reference and detected trees
ref.trees <- cbind(c(1,4,3,4,2), c(1,1.5,2,3,4), c(15,18,20,10,11))
def.trees <- cbind(c(2,2,4,4), c(1,3,4,1), c(16,19,9,15))

# compute matching
match1 <- treeMatching(ref.trees, def.trees)
match2 <- treeMatching(ref.trees, def.trees, deltaGround=2, hPrec=0)

# 2D display of matching results
plot2Dmatched(ref.trees, def.trees, match1, xlab="X", ylab="Y")
plot2Dmatched(ref.trees, def.trees, match2, xlab="X", ylab="Y")
```

plotTreeInventory

 Displays a map of tree inventory data

Description

displays tree inventory data

Usage

```r
plotTreeInventory(xy, height = NULL, diam = NULL, species = NULL, ...)
```

Arguments

- `xy` data.frame. contains two columns with the X, Y coordinates of tree centers
- `height` vector. tree heights in meters
- `diam` vector. tree diameters in centimeters
- `species` vector. species abbreviation as in `speciesColor` for display with corresponding color
- `...` Arguments to be passed to methods, as in `plot`

Value

no return

See Also

- `speciesColor` for a table of species and associated colors
Examples

# load tree inventory data from plot Chablais 3
data("treeinventorychablais3")

# display tree inventory
plotTreeInventory(treeinventorychablais3[,c("x","y")],
diam = treeinventorychablais3$d, col = "red",
pch = treeinventorychablais3$e,
xlab = "X", ylab = "Y")

# display tree inventory with CHM background
data("chmchablais3")
raster::plot(chmchablais3, col=gray(seq(0,1,1/255)))
plotTreeInventory(treeinventorychablais3[,c("x","y")],
height = treeinventorychablais3$h,
species = treeinventorychablais3$s,
add = TRUE)

pointList2SPDF

Convert list of points into Spatial Polygons DataFrame object

Description

Converts a list of points specifying polygons into a Spatial Polygons DataFrame object

Usage

pointList2SPDF(points.list, df = NULL, ...)

Arguments

points.list list of dataframes of xy coordinates. The first and last coordinates in each
dataframe must be the same
df data.frame. Optional data.frame to be associated to Spatial Polygons
... arguments to be passed to SpatialPolygons

Value

an object of class SpatialPolygons-class, or class SpatialPolygonsDataFrame-class if input data.frame is specified.

See Also

ellipses4Crown
Examples

# Compute coordinates of polygons
ellipses <- ellipses4Crown(c(0,10), c(0,10), c(2,2), c(3,4), c(2.5,3), c(2,3),
id=c("A", "B"))

# Convert to Spatial object
ellipses1 <- pointList2SPDF(ellipses)

# Convert to Spatial object with data.frame
ellipses2 <- pointList2SPDF(ellipses, df=data.frame(info=1:2))

# draw ellipses
sp::plot(ellipses2, col = ellipses2$info)

points2DSM

Digital Surface Model

Description

Creates a Digital Surface Model from LAS object. Raster extent is specified by the coordinates of lower left and upper right corners. Default extent covers the full range of points, and aligns on multiple values of the resolution. Cell value is the maximum height of points contained in the cell.

Usage

points2DSM(.las, res = 1, xmin, xmax, ymin, ymax)

Arguments

<table>
<thead>
<tr>
<th>.las</th>
<th>LAS object or XYZ matrix/data.frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>res</td>
<td>numeric. raster resolution</td>
</tr>
<tr>
<td>xmin, xmax</td>
<td>numeric. lower left corner easting coordinate for output raster.</td>
</tr>
<tr>
<td>ymin, ymax</td>
<td>numeric. upper right corner easting coordinate for output raster.</td>
</tr>
</tbody>
</table>

Value

A raster object.

See Also

points2DTM for Digital Terrain Model computation.
Examples

```r
data(laschablais3)

# create a digital surface model with first-return points, resolution 0.5 m
dsm <- points2DSM(lidR::filter_first(laschablais3), res=0.5)

# display raster
raster::plot(dsm,asp=1)
```

---

**points2DTM**

*Digital Terrain Model*

Description

Creates a Digital Terrain Model from LAS object or XYZ data. Raster extent is specified by the coordinates of lower left and upper right corners. Default extent covers the full range of points, and aligns on multiple values of the resolution. Cell value is compute as the bilinear interpolation at the cell center form an Delaunay triangulation. Relies on `grid_terrain` with algorithm `tin`. In case a LAS object is provided, only points classified as ground or water (2 or 9) will be used; a warning is issued when the raster output inherits the projection info from LAS input.

Usage

```r
points2DTM(.las, res = 1, xmin, xmax, ymin, ymax)
```

Arguments

- `.las` LAS object or XYZ matrix/data.frame containing only ground points
- `res` numeric. raster resolution
- `xmin` numeric. lower left corner easting coordinate for output raster.
- `xmax` numeric. upper right corner easting coordinate for output raster.
- `ymin` numeric. lower left corner northing coordinate for output raster.
- `ymax` numeric. upper right corner northing coordinate for output raster.

Value

A raster object

See Also

`points2DSM` for Digital Surface Model computation.
Examples

```r
data(laschablais3)

# create digital terrain model with points classified as ground
dtm <- points2DTM(laschablais3)

# display raster
raster::plot(dtm,asp=1)
```

points2terrainStats  Computation of terrain metrics

Description

This function computes topographic variables from a point cloud

- exposition
- altitude
- slope

values are computed after fitting a plane to the points. It supposes a homogeneous sampling of the plot by points. Points can be cropped on disk if center and radius are provided. In case a centre is provided, the altitude is computed by bilinear interpolation at the center location (`grid_metrics` with tin algorithm), otherwise it is the mean of the points altitude range.

Usage

```r
points2terrainStats(p, centre = NULL, r = NULL)
```

Arguments

- `p`  
matrix, data.frame or LAS object with ground point coordinates (X, Y, Z). In case of an object which is not LAS, the object is first converted, which issues a warning

- `centre`  
vector. x y coordinates of center to extract points inside a disc

- `r`  
numeric. radius of disc

Value

a data.frame with altitude, exposition (gr), slope (gr) and adjR2 of plane fitting
Examples

```r
# sample points
XYZ <- data.frame(x=runif(200, -10, 10), y=runif(200, -10, 10))
XYZ$z <- 350 + 0.3 * XYZ$x + 0.1 * XYZ$y + rnorm(200, mean=0, sd=0.5)
# compute terrain statistics
points2terrainStats(XYZ)
points2terrainStats(XYZ, centre=c(5,5), r=5)
# with a LAS object
data(laschablais3)
terrain.points <- lidR::filter_ground(laschablais3)
points2terrainStats(terrain.points)
points2terrainStats(terrain.points, centre=c(974360, 6581650), r=10)
```

---

### polar2Projected

**Polar to cartesian coordinates conversion**

**Description**

Computes projected coordinates (Easting, Northing, Altitude) from polar coordinates (Azimuth, Slope, Distance) and center position (Easting, Northing, Altitude). Magnetic declination and meridian convergence are optional parameters. In case distance is measured to the border of objects (e.g. trees), the diameter can be added to compute the coordinates of object center.

**Usage**

```r
polar2Projected(
  x,  # vector. easting coordinates of centers in meter
  y,  # vector. northing coordinates of centers in meter
  z = 0,  # vector. altitudes of centers in meters
  azimuth,  # vector. azimuth values from centers in radian
  dist,  # vector. distances between centers and objects in meter
  slope = 0,  # vector. slope values from centers in radian
  declination = 0,  # vector. magnetic declination values in radian
  convergence = 0,  # vector. meridian convergence values in radian
  diameter = 0  # vector. diameters in meter (e.g. in case a radius should be added to the distance)
)
```

**Arguments**

- `x`, `y`, `z`:
  - `x`:
    - Vector of easting coordinates of centers in meter.
  - `y`:
    - Vector of northing coordinates of centers in meter.
  - `z`:
    - Vector of altitudes of centers in meters.
- `azimuth`:
  - Vector of azimuth values from centers in radian.
- `dist`:
  - Vector of distances between centers and objects in meter.
- `slope`:
  - Vector of slope values from centers in radian.
- `declination`:
  - Vector of magnetic declination values in radian.
- `convergence`:
  - Vector of meridian convergence values in radian.
- `diameter`:
  - Vector of diameters in meter (e.g. in case a radius should be added to the distance).
quatremontagnes

Value

A data.frame with easting, northing and altitude coordinates, and horizontal distance from centers to objects centers

See Also

plotTreeInventory for tree inventory display

Examples

# create data.frame of trees with polar coordinates and diameters
trees <- data.frame (x=rep(c(0,10), each=2),
y = rep(c(0,10),each=2),
z=rep(c(0,2),each=2),
azimuth=rep(c(0,pi/3)),
dist=rep(c(2,4)),
slope=rep(c(0,pi/6)),
diameter.cm=c(15,20,25,30))
trees

# compute projected coordinates
polar2Projected(trees$x, trees$y, trees$z, trees$azimuth, trees$dist,
trees$slope, declination=0.03, convergence=0.02, trees$diameter.cm/100)

quatremontagnes  
Field plot inventory in the Quatre Montagnes area (France)

Description

Dataset of forest parameters measured in the field on 96 circular plots of 15 m radius. Metrics derived from airborne laser scanning (ALS) point clouds have also been extracted and calculated for those plots.

Usage

data(quatremontagnes)

Format

A data.frame with 76 columns:
1. plotId id of field plot
2. X easting coordinate (epsg: 2154)
3. Y northing coordinate (epsg: 2154)
4. clusterId id of cluster, plots were inventoried in groups of 4
5. G.m2.ha basal area in m2 per ha
6. N.ha number of trees per ha
raster2Cimg

7. \textit{D.mean.cm} mean tree diameter at breast height (1.3 m) in cm
8. \textit{stratum} forest ownership (public or private)
9. [,9:60] point cloud metrics computed from ALS, see \texttt{ABAmodelMetrics}
10. [,6:73] metrics derived from tree segmentation in ALS data, see \texttt{stdTreeMetrics}
11. [,73:76] terrain statistics, see \texttt{points2terrainStats}

References

Monnet, J.-M. 2021. Tutorial on modeling forest parameters with ALS data. ABA data preparation

Examples

data(quatremontagnes)
summary(quatremontagnes)

---

raster2Cimg \hspace{1cm} \textit{RasterLayer to Cimg conversion}

Description

converts a \texttt{RasterLayer} object to \texttt{Cimg} object. NA values in raster are replaced.

Usage

raster2Cimg(rasterLayer, NA.replace = 0, maxpixels = 1e+10)

Arguments

rasterLayer \hspace{1cm} \texttt{rasterLayer} \hspace{1cm} \texttt{rasterLayer} object. raster of canopy height model, preferably filtered to avoid effect of holes on volume and surface computation

NA.replace \hspace{1cm} \texttt{numeric} \hspace{1cm} \texttt{numeric} value to replace NA values with.

maxpixels \hspace{1cm} \texttt{numeric} \hspace{1cm} maximum number of pixels to be converted to cimg (argument passed to \texttt{as.cimg}).

Value

A \texttt{cimg} object

See Also

cimg2Raster
Examples

data(chmchablais3)

chm.cim <- raster2Cimg(chmchablais3)
chm.cim
summary(chm.cim)

# plot rasterLayer
raster::plot(chmchablais3)

# plot cimg object
plot(chm.cim)

rasterChullMask  Raster mask of convex hull

Description

creates raster mask corresponding to the convex hull of xy positions

Usage

rasterChullMask(xy, r)

Arguments

xy  2 columns matrix or data.frame. xy positions
r   raster object. target raster

Value

a raster with 0 or 1

See Also

rasterXYMask

Examples

# create raster
r <- raster::raster()
raster::extent(r) <- c(0,40,0,40)
raster::res(r) <- 1

# xy positions
xy <- data.frame(c(10,20,31.25,15),
                 c(10,20,31.25,25))

# compute mask
mask1 <- rasterChullMask(xy, r)
rasterLocalmax

Description

identifies global maximum and second global maximum from raster (e.g. output from `rastersMovingCor`), and computes related statistics. Local maxima can be excluded based on a minimum distance $d_m$ to nearest local maximum.

Usage

```r
rasterLocalmax(r, dm = 2, med1 = 1, med2 = 2, quanta = 0.75, quantb = 0.5)
```

Arguments

- `r` : raster. typically output of `rastersMovingCor`
- `dm` : numeric. minimum distance between two local maxima in meters
- `med1` : numeric. window radius to compute median value around the maximum position (default: 1m)
- `med2` : numeric. window radius #2 to compute median value around the maximum position (default: 2m)
- `quanta` : numeric. quantile value to compute for raster values (default: 3rd quartile)
- `quantb` : numeric. quantile #2 value to compute for raster values (default: median)

Value

A data.frame with value of maximum, position of maximum, position of second maximum, ratio of max value to 2nd max, ratio of max value to median of neighborhood (size1 and size 2), ratio of max value to raster quantiles 1 and 2

See Also

`rastersMovingCor`, `coregistration` for application to the coregistration of tree inventory data with canopy height models
# create raster
r.b <- raster::raster()
raster::extent(r.b) <- c(0,40,0,40)
raster::res(r.b) <- 1
xy <- raster::xyFromCell(r.b,1:length(r.b))

# add Gaussian surfaces
z1 <- 1.5*exp(-(xy[,1]-22)^2+(xy[,2]-22)^2/2)/5
z2 <- exp(-(xy[,1]-20)^2+(xy[,2]-22)^2/2)/3
z3 <- 1.5*exp(-(xy[,1]-17)^2+(xy[,2]-17)^2/2)/5
r.b <- raster::rasterFromXYZ(cbind(xy,z1+z2+z3))

# create small raster
r.s <- raster::crop(r.b, raster::extent(c(15,25,15,25)))
# offset raster by (-2, -2)
raster::extent(r.s) <- c(13,23,13,23)

rr <- rastersMovingCor(r.b, r.s, buffer=6, step=1)
loc.max <- rasterLocalmax(rr)
loc.max

# plot raster
raster::plot(rr)
# add location of two local maxima
graphics::points(loc.max[1,c("dx1","dx2")], loc.max[1,c("dy1","dy2")],
               cex=c(1,0.5), pch=3)

---

**rasterMetrics**

*Computes metrics by aggregating a raster at lower resolution or summarizing attributes based on XY locations*

**Description**

Compute statistics by aggregating a raster at lower resolution. Aggregation groups are larger cells, new values are computed by applying a user-specified function to original cells contained in the larger cells. Results are provided as a data.frame which also contains the XY coordinates of the larger cells.

**Usage**

rasterMetrics(
  r,
  res = 20,
  fun = function(x) { data.frame(mean = mean(x$layer), sd = stats::sd(x$layer)) },
  output = "raster"
)
Arguments

r raster object or data.frame with xy coordinates in two first columns
res numeric. Resolution of the aggregation raster, should be a multiple of r resolution if a raster is provided
fun function. Function to compute metrics in each aggregated cell from the values contained in the initial raster (use x$layer to access raster values) / data.frame (use x$column_name to access values)
output string. indicates the class of output object "raster" or "data.frame"

Value

a data.frame with the XY center coordinates of the aggregated cells, and the values computed with the user-specified function or a Raster object

Examples

data(chmchablais3)

# raster metrics from raster
metrics1 <- rasterMetrics(chmchablais3, res=10)
metrics1

# raster metrics from data.frame
n <- 1000
df <- data.frame(x=runif(n, 0,100), y=runif(n, 0, 100), z1=runif(n, 0,1), z2=runif(n,10,20))
# compute raster metrics
metrics2 <- rasterMetrics(df, res=10, fun=function(x){data.frame(max.z=max(x$z1), max.sum=max(x$z1+x$z2))}, output="data.frame")
summary(metrics2)

# display raster metrics
raster::plot(metrics1)
# display data.frame metrics
raster::plot(raster::rasterFromXYZ(metrics2))

rasters2Cor Correlation between two rasters

Description

computes correlation between two rasters, based on the extent of the smallest one.

Usage

rasters2Cor(raster.b, raster.s, mask = NULL, small.SC = TRUE)
Arguments

- \texttt{raster.b} \hspace{1em} raster, raster to correlate with largest extent
- \texttt{raster.s} \hspace{1em} raster, raster to correlate with smallest extent
- \texttt{mask} \hspace{1em} raster, mask of area to correlate
- \texttt{small.SC} \hspace{1em} boolean, is the small raster already standardized and centered?

Value

A numeric

See Also

- \texttt{rastersMovingCor} to compute correlation between rasters for different translations

Examples

```r
# create raster
r.b <- raster::raster()
raster::extent(r.b) <- c(0,40,0,40)
raster::res(r.b) <- 1
xy <- raster::xyFromCell(r.b, 1:length(r.b))

# add Gaussian surface and noise
z <- 3*exp(-((xy[,1]-20)^2+(xy[,2]-20)^2/2)/6)
r.b <- raster::rasterFromXYZ(cbind(xy, z))

# create circular mask of radius 5
z.mask <- (xy[,1]-20)^2 + (xy[,2]-20)^2 < 5^2
r.mask <- raster::rasterFromXYZ(cbind(xy, z.mask))

# create small raster of size 20
r.s <- raster::crop(r.b, raster::extent(c(10,30,10,30)))

# add noise to small raster
raster::values(r.s) <- raster::values(r.s) + rnorm(length(r.s), 0, 0.5)
r.mask <- raster::crop(r.mask, raster::extent(c(10,30,10,30)))

# compute correlation on masked area where signal to noise ratio is lower
rasters2Cor(r.b, r.s, r.mask, small.SC=FALSE)

# compute correlation for whole small raster
rasters2Cor(r.b, r.s, small.SC=FALSE)

# display large raster
raster::plot(r.b, main="Large raster")
# display small raster
raster::plot(r.s, main="Small raster")
# display mask
raster::plot(r.mask, main="Computation mask")
```
**rastersMovingCor**  
*Correlation between rasters for different XY translations*

**Description**

computes correlation between two rasters for different XY translations. The correlation values are computed on the extent of the smallest raster using `rasters2Cor`, after applying an optional mask, and for each translation within a buffer area.

**Usage**

```r
rastersMovingCor(raster.b, raster.s, mask = NULL, buffer = 19, step = 0.5)
```

**Arguments**

- `raster.b`: raster to correlate with largest extent
- `raster.s`: raster to correlate with smallest extent
- `mask`: raster mask of area to correlate, applied to small raster
- `buffer`: numeric. radius of the circular buffer area for possible translations
- `step`: numeric. increment step of translations within buffer area to compute correlation values, should be a multiple of raster resolution

**Value**

A raster. Raster value at coordinates x,y correspond to the correlation between the large raster and the small raster when small raster center has been translated of (x,y)

**See Also**

- `rasterLocalmax` to extract local maximum of resulting correlation raster, `rasters2Cor`

**Examples**

```r
# create raster
r.b <- raster::raster()
raster::extent(r.b) <- c(0,40,0,40)
raster::res(r.b) <- 1
xy <- raster::xyFromCell(r.b,1:length(r.b))

# add Gaussian surfaces
z1 <- 1.5*exp(-((xy[,1]-22)^2+(xy[,2]-22)^2/2)/5)
z2 <- exp(-((xy[,1]-20)^2+(xy[,2]-22)^2/2)/3)
z3 <- 1.5*exp(-((xy[,1]-17)^2+(xy[,2]-17)^2/2)/5)
r.b <- raster::rasterFromXYZ(cbind(xy,z1+z2+z3))

# create small raster
r.s <- raster::crop(r.b, raster::extent(c(15,25,15,25)))
# offset raster by (-2, -2)
```
raster::extent(r.s) <- c(13,23,13,23)

# compute correlations for translations inside buffer
rr <- rastersMovingCor(r.b, r.s, buffer=6, step=1)
rr

# display large raster
raster::plot(r.b, main="Large raster")
# display small raster
raster::plot(r.s, main="Small raster")
# display correlation
raster::plot(rr, xlab="X translation", ylab="Y translation",
main="Correlation between rasters")

---

**rasterXYMask**

*Raster mask by union of buffers around xy positions*

**Description**

creates a raster mask by union of circular buffers around xy positions

**Usage**

rasterXYMask(xy, buff, r, binary = TRUE)

**Arguments**

- **xy**: 2 columns matrix or data.frame. xy positions
- **buff**: vector. buffers to apply to the xy positions
- **r**: raster object. target raster
- **binary**: boolean. should the output mask be boolean (TRUE) or greyscale (FALSE)

**Value**

a raster object

**See Also**

rasterChullMask

**Examples**

# create raster
r <- raster::raster()
raster::extent(r) <- c(0,40,0,40)
raster::res(r) <- 1

# xy positions
rasterZonalStats <- data.frame(x=c(10,20,31.25,15),
y=c(10,20,31.25,25))
# compute mask
mask1 <- rasterXYMask(xy, c(5, 8, 5, 5), r)
mask2 <- rasterXYMask(xy, c(5, 8, 5, 5), r, binary=FALSE)

# display binary raster
raster::plot(mask1)
graphics::points(xy)

# display distance raster
raster::plot(mask2)
graphics::points(xy)

rasterZonalStats  Image statistic in segment

Description
compute zonal statistic of an image

Usage
rasterZonalStats(segments, dem.nl, fun = max)

Arguments
- segments: cimg or rasterLayer object. image with segments id (e.g. from segmentation)
- dem.nl: cimg or rasterLayer object. image to compute statistic from
- fun: function to compute statistic from values in each segment

Value
A cimg object or raster object with values of the statistic

See Also
segmentation

Examples
data(chmchablais3)

# median filter
chmchablais3 <- demFiltering(chmchablais3, nlFilter="Median", nlSize=3,
sigma=0)$non.linear.image

# maxima detection
maxi <- maximaDetection(chmchablais3)
# segmentation
seg.maxi <- segmentation(maxi, chmchablais3)

# compute image of maximum value in each segment
max.in.segment <- rasterZonalStats(seg.maxi, chmchablais3)

# plot original image
raster::plot(chmchablais3, main="Median filter")

# plot segments and image of max value inside segments
seg.maxi[seg.maxi==0] <- NA
raster::plot(seg.maxi %% 8, main="Segments", col=rainbow(8))
raster::plot(max.in.segment, main="Max value in segment")

segAdjust \hspace{1cm} Modification of segments based on values

Description

in a segmented image, removes from segments the pixels which values in a reference image is below a certain percentage of the highest value inside the segment. Removed pixels are attributed 0 value.

Usage

segAdjust(dem.w, dem.wh, dem.nl, prop = 0.3, min.value = 2, min.maxvalue = 5)

Arguments

dem.w \hspace{1cm} cimg or rasterLayer object. image with segments id, without 0 values
dem.wh \hspace{1cm} cimg or rasterLayer object. image with max value inside segment
dem.nl \hspace{1cm} cimg or rasterLayer object. image with initial values
prop \hspace{1cm} numeric. proportional threshold for removal of pixels which initial values are lower than the max height of the segment (dem.nl < prop x dem.wh)
min.value \hspace{1cm} numeric. threshold for removal of pixels which initial values are lower (dem.nl < min.value)
min.maxvalue \hspace{1cm} numeric. threshold for complete removal of segments which maximum value height is smaller to the threshold (dem.wh < min.maxvalue)

Value

A cimg or rasterLayer object: image with modified segments.

See Also

maximaDetection, maximaSelection
Examples

```r
data(chmchablais3)

# median filter
chmchablais3 <- demFiltering(chmchablais3, nlFilter="Median", nlSize=3, sigmap=0)$non.linear.image

# maxima detection and selection
maxi <- maximaDetection(chmchablais3)
selected.maxi <- maximaSelection(maxi, chmchablais3, dm=1, dprop=0.1)

# segmentation
seg.selected.maxi <- segmentation(selected.maxi, chmchablais3)

# max value in segments
max.in.segment <- rasterZonalStats(seg.selected.maxi, chmchablais3)

# segmentation modification
seg.modif1 <- segAdjust(seg.selected.maxi, max.in.segment, chmchablais3, prop=0.5)
seg.modif2 <- segAdjust(seg.selected.maxi, max.in.segment, chmchablais3, prop=0, min.value=5, min.maxvalue=10)

# plot initial segmented image
seg.selected.maxi[seg.selected.maxi==0] <- NA
raster::plot(seg.selected.maxi %% 8, main="Initial segments", col=rainbow(8))
seg.modif1[seg.modif1==0] <- NA
raster::plot(seg.modif1 %% 8, main="Modified segments 1", col=rainbow(8))
seg.modif2[seg.modif2==0] <- NA
raster::plot(seg.modif2 %% 8, main="Modified segments 2", col=rainbow(8))
```

---

**segmentation**

*Image segmentation by seed-based watershed algorithm*

**Description**

performs a seed-based watershed segmentation (wrapper for imager::watershed)

**Usage**

```r
segmentation(maxi, dem.nl)
```

**Arguments**

- `maxi` cimg or rasterLayer object. image with seed points (e.g. from `maximaDetection` or `maximaSelection`)
- `dem.nl` cimg or rasterLayer object. image for seed propagation (typically initial image used for maxima detection).
speciesColor

Value

A cimg object or rasterLayer object with segments id

See Also

maximaDetection, maximaSelection, segAdjust

Examples

data(chmchablais3)

# median filter
chmchablais3 <- demFiltering(chmchablais3, nlFilter="Median", nlSize=3, 
   sigmap=0)$non.linear.image

# maxima detection
maxi <- maximaDetection(chmchablais3)

# maxima selection
selected.maxi <- maximaSelection(maxi, chmchablais3, dm=1, dprop=0.1)

# segmentation
seg.maxi <- segmentation(maxi, chmchablais3)
seg.selected.maxi <- segmentation(selected.maxi, chmchablais3)

# plot original image
raster::plot(chmchablais3, main="Median filter")

# plot segmented image
# replace segment with id 0 (not a tree) with NA
seg.maxi[seg.maxi==0] <- NA
raster::plot(seg.maxi %% 8, main="Segments, no maxima selection", col=rainbow(8))
seg.selected.maxi[seg.selected.maxi==0] <- NA
raster::plot(seg.selected.maxi %% 8, main="Segments, maxima selection", col=rainbow(8))


<table>
<thead>
<tr>
<th>speciesColor</th>
<th>Table of species names, abbreviations and display colors</th>
</tr>
</thead>
</table>

Description

table for species names, abbreviations and type (coniferous/broadleaf), and display color

Usage

speciesColor()
stdTreeMetrics

Description

This function computes summary statistics from a data.frame containing tree-level information as returned by `treeExtraction`.

Usage

```
stdTreeMetrics(x, area.ha = NA)
```

Arguments

- `x`: data.frame containing the following columns for each line (segmented tree): `h` (height), `s` (crown surface), `v` (crown volume), typically returned by `treeExtraction`. `sp` (crown surface inside region of interest) and `vp` (crown volume in region of interest) are not used in this function.
- `area.ha`: numeric. area of region of interest in ha

Value

a data.frame with one line containing the following tree metrics:

1. `Tree.meanH`: mean height of detected tree apices (m)
2. `Tree.sdH`: standard deviation of heights of detected tree apices (m)
3. `Tree.giniH`: Gini index of heights of detected tree apices
4. `Tree.density`: density of detected tree apices (/ha)
5. `TreeInf10.density`: density of detected trees apices with h<=10 (/ha)
6. `TreeSup10.density`: density of detected trees apices with h>10 (/ha)
7. `TreeSup20.density`: density of detected trees apices with h>20 (/ha)
8. TreeSup30.density: density of detected trees apices with h>30 (/ha)
9. Tree.meanCrownSurface: mean crown surface of detected trees
10. Tree.meanCrownVolume: mean volume of detected trees
11. TreeCanopy.meanH: mean height of union of crowns of detected trees

See Also

treeExtraction

Examples

```r
# sample 50 height values
h <- runif(50, 5, 40)
# simulate tree data.frame
trees <- data.frame(h=h, s=h, sp=h*0.95, v=h*h*0.6, vp=h*h*0.55)
stdTreeMetrics(trees, area.ha=0.1)
```

Description

creates a dataframe with segment id, height and coordinates of maxima, surface and volume, computed from three images: initial, local maxima and segmented, obtained with treeSegmentation

Usage

treeExtraction(r.dem.nl, r.maxi, r.dem.w, r.mask = NULL)

Arguments

- `r.dem.nl`: raster object. raster of canopy height model, preferably filtered to avoid effect of holes on volume and surface computation
- `r.maxi`: raster object. raster positive values at local maxima
- `r.dem.w`: raster object. segmented raster
- `r.mask`: raster object. only segments which maxima are inside the mask are extracted

Value

A spatial data.frame with tree id, local maximum stats (height, dominance radius), segment stats (surface and volume).

See Also

treeSegmentation
Examples

data(chmchablais3)

# tree segmentation
segments <- treeSegmentation(chmchablais3)

# tree extraction
trees <- treeExtraction(segments$filled.dem, segments$local.maxima,
                        segments$segments.id)
trees

# plot initial image
raster::plot(chmchablais3)

# add treetop positions
sp::plot(trees, cex=trees$h/20, add=TRUE, pch=1)

# add segment contours (vectorization is slow)
contours <- raster::rasterToPolygons(segments$segments.id, dissolve=TRUE)
sp::plot(contours, add=TRUE, border="white")

--

treeinventorychablais3

Tree inventory data in France (Chablais 3 plot, July 2010)

Description

All trees with diameter at breast height >= 7.5 cm are inventoried on a 50m x 50m plot.

Usage

data(treeinventorychablais3)

Format

A data.frame with columns:

1. x easting coordinate (epsg: 2154)
2. y northing coordinate (epsg: 2154)
3. d dbh (cm)
4. h tree height (m)
5. n tree number
6. s species abbreviated as GESP (GEnus SPecies)
7. e appearance (0: missing or lying, 1: normal, 2: broken treetop, 3: dead with branches, 4: snag)
8. t tilted (0: no, 1: yes)
References


Examples

data(treeinventorychablais3)
summary(treeinventorychablais3)

---

treeMatching

3D matching of detected tree top positions with reference positions

Description

First computes a matching index for each potential pair associating a detected with a reference tree. This index is the 3D distance between detected and reference points, divided by a maximum matching distance set by user-defined parameters. Pairs with the lowest index are then iteratively associated.

Usage

treeMatching(lr, ld, deltaGround = 2.1, hPrec = 0.14, computeStat = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lr</td>
<td>data.frame or matrix. 3D coordinates (X Y Height) of reference positions</td>
</tr>
<tr>
<td>ld</td>
<td>data.frame or matrix. 3D coordinates (X Y Height) of detected positions</td>
</tr>
<tr>
<td>deltaGround</td>
<td>numeric. buffer around trunk position: absolute value</td>
</tr>
<tr>
<td>hPrec</td>
<td>numeric. buffer around apex position: proportion of reference tree height</td>
</tr>
<tr>
<td>computeStat</td>
<td>boolean. should matching stats be computed</td>
</tr>
</tbody>
</table>

Value

A data.frame with matched pairs (row of reference positions in first column, and row of detected positions in second column) and corresponding 3D distances

References


See Also

plot2Dmatched, histDetection

Examples

# create reference and detected trees
ref.trees <- cbind(c(1,4,3,4,2), c(1,1,2,3,4), c(15,18,20,10,11))
def.trees <- cbind(c(2,2,4,4), c(1,3,4,1), c(16,19,9,15))
#
# match trees
match1 <- treeMatching(ref.trees, def.trees)
match2 <- treeMatching(ref.trees, def.trees, deltaGround=2, hPrec=0)
match1
match2

# 2D display of matching result
plot2Dmatched(ref.trees, def.trees, match1, xlab="X", ylab="Y")
plot2Dmatched(ref.trees, def.trees, match2, xlab="X", ylab="Y")
treeSegmentation

nlFilter  string. specifies the non-linear filter for image pre-processing, should be an option of function demFiltering

nlSize  numeric. width of kernel of non-linear filter in pixels

sigma  numeric or matrix. if a single number is provided, sigma is the standard deviation of Gaussian filter in meters, 0 corresponds to no smoothing. In case of matrix, the first column corresponds to the standard deviation of the filter, and the second to thresholds for image values (e.g. a filter of standard deviation specified in line i is applied to pixels in image which values are between thresholds indicated in lines i and i+1). Threshold values should be ordered in increasing order.

dmin  numeric. treetop minimum distance to next higher pixel in meters

dprop  numeric. number defining the treetop minimum distance as proportion of height to next higher pixel

hmin  numeric. minimum treetop height

crownProp  numeric. minimum height of tree crown as proportion of treetop height

crownMinH  numeric. minimum crown height

dtm  raster object or string indicating location of raster file with the terrain model. If provided, the maxima extraction and watershed segmentation are performed on the dem (this avoids the deformation of crown because of the normalisation with terrain), but maxima selection and segment adjustment are performed on 'dem-dtm' because the selection criteria is the height to terrain.

Value

A RasterStack with 4 layers: selected local maxima (values = distance to higher pixel), segments, non-linear preprocessed dem, smoothed preprocessed dem

References


See Also
demFiltering, maximaDetection, maximaDetection, maximaSelection, segmentation, segAdjust, treeExtraction
Examples

```r
data(chmchablais3)

# tree segmentation
segments <- treeSegmentation(chmchablais3)
segments2 <- treeSegmentation(chmchablais3, nlFilter="Median", nlSize=3,
   sigma=cbind(c(0.2,0.8), c(0,15)), dmin=0, dprop=0, hmin=10, crownProp=0.5, crownMinH=5)

# plot initial image segments
raster::plot(chmchablais3, main="Initial image")
raster::plot(segments$smoothed.dem, main="Filtered image")
raster::plot(segments$local.maxima, main="Local maxima")

# replace segment with id 0 (not a tree) with NA
segments$segments.id[segments$segments.id==0] <- NA
raster::plot(segments$segments.id %% 8, main="Segments", col=rainbow(8))

# plot segmentation with other parameters
segments2$segments.id[segments2$segments.id==0] <- NA
raster::plot(segments2$segments.id %% 8, main="Segments2", col=rainbow(8))
```
Index

* datasets
  ABAmodelMetrics, 7
  chmchablais3, 11
  laschablais3, 30
  quatremontagnes, 41
  treeinventorychablais3, 56
  .ABAmodelMetrics (ABAmodelMetrics), 7

ABAinference, 3
ABAmodel, 3, 4, 6, 8, 9
ABAmodelCombineStrata, 3, 5, 5, 9
ABAmodelMetrics, 7, 15, 42
ABAmodelPlot, 5, 6, 8
ABApredict, 9
AddVegetationIndices, 10
as.cimg, 20

BoxcoxTr, 10, 28, 29

chmchablais3, 11
cimg, 32
cimg2Raster, 12, 42
circle2Raster, 13
cleanRaster, 9, 14
cloud_metrics, 7, 15
cloudMetrics, 7, 15
cloudTreeMetrics, 16
coregistration, 17, 44
creatDisk, 19

demFiltering, 20, 23, 24, 32, 59
deriche, 20

detection, 21, 24
detection, 21, 23
detection, 21, 23
detection, 21, 23
detection, 21, 23
detection, 21, 23
detection, 21, 23

gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23
gapDetection, 21, 23

glm, 5, 6, 31
glm, 5, 6, 31

heightRegression, 25
histDetection, 26, 34, 58
histStack, 27
iBoxcoxTr, 11, 28, 29
iBoxcoxTrBiasCor, 5, 11, 28, 29

LAS, 15, 16, 30, 37–39
laschablais3, 30
lm, 5, 6, 31
lm, 5, 6, 31
lmaCheck, 5, 31
maximaDetection, 20, 32, 33, 51–53, 59
maximaDetection, 20, 32, 33, 51–53, 59
maximaSelection, 32, 33, 51–53, 59
maximaSelection, 32, 33, 51–53, 59
mclosing, 20
medianBlur, 20

plot, 8, 25, 27, 34, 35
plot2Dmatched, 34, 58
plotTreeInventory, 35, 41, 54
pointList2SPDF, 23, 36
points2DSM, 16, 37, 38
points2DTM, 37, 38
points2terrainStats, 39, 42
polar2Projected, 40

quatremontagnes, 41

raster, 20, 32
raster2Cimg, 12, 42
rasterChullMask, 43, 49
rasterLocalmax, 18, 44, 48
rasterMetrics, 45
rasters2Cor, 46, 48
rastersMovingCor, 18, 44, 47, 48
rasterXYMask, 43, 49
rasterZonalStats, 50
regsubsets, 4, 5
res, 32

segAdjust, 51, 53, 59
segmentation, 50, 52, 59
SpatialPolygons, 36
SpatialPolygons-class, 36
SpatialPolygonsDataFrame-class, 36
speciesColor, 25, 35, 53
stdmetrics, 7, 15
stdmetrics_i, 7
stdmetrics_z, 7
stdTreeMetrics, 16, 42, 54

tin, 38, 39
treeExtraction, 16, 54, 55, 55, 59
treeinventorychablais, 56
treeMatching, 25, 26, 34, 57
treeSegmentation, 16, 55, 58

vif, 31