Package 'lidaRtRee'

January 23, 2025

Type Package

Version 4.0.8

Title Forest Analysis with Airborne Laser Scanning (LiDAR) Data

Date 2025-01-22

Description Provides functions for forest objects detection, structure metrics computation, model calibration and mapping with airborne laser scanning: co-registration of field plots (Monnet and Mermin (2014) <doi:10.3390/f5092307>); tree detection (method 1 in Eysn et al. (2015) <doi:10.3390/f6051721>) and segmentation; forest parameters estimation with the area-based approach: model calibration with ground reference, and maps export (Aussenac et al. (2023) <doi:10.12688/openreseurope.15373.2>); extraction of both physical (gaps, edges, trees) and statistical features useful for e.g. habitat suitability modeling (Glad et al. (2020) <doi:10.1002/rse2.117>) and forest maturity mapping (Fuhr et al. (2022) <doi:10.1002/rse2.274>).

URL `https://lidar.pages.mia.inra.fr/lidaRtRee/`,
`https://forgemia.inra.fr/lidar/lidaRtRee`

BugReports `https://forgemia.inra.fr/lidar/lidaRtRee/-/issues`

Depends R (>= 4.0.0)

Imports graphics, stats, methods, grDevices, terra, sf, imager, leaps, gvlma, car, reldist, lidR (>= 4.1.1)

Suggests stars

License GPL-3

LazyData TRUE

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Jean-Matthieu Monnet [aut, cre] (<`https://orcid.org/0000-0002-9948-9891`>),
Pascal Obstétar [ctb] (<`https://orcid.org/0000-0002-2811-7548`>)

Maintainer Jean-Matthieu Monnet <jean-matthieu.monnet@inrae.fr>

Repository CRAN

Date/Publication 2025-01-23 09:40:02 UTC

Contents

`aba_build_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

```
aba_build_model(
  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. Non-finite values are replaced by the minimum. Values are replaced after computation of the variance of residuals used for bias correction
`...`	other parameters to be passed to `regsubsets`,

Value

a list with three elements

- `model`: list with one regression model (output from `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

`aba_combine_strata` for combining models calibrated on different strata, `aba_plot` for plotting model cross-validation results, `regsubsets` for variable selection, `lma_check` for linear model assumptions check, `boxcox_itr_bias_cor` for reverse Box-Cox transformation with bias correction.

Examples

```
data(quatre_montagnes)
# build ABA model for basal area, with all metrics as predictors
model_aba <- aba_build_model(quatre_montagnes$G_m2_ha, quatre_montagnes[, 9:76],
  transform = "boxcox", nmax = 2
)
# summary of regression model
summary(model_aba$model)
# validation statistics
model_aba$stats
```

```
# observed and predicted values
summary(model_aba$values)

# plot field values VS predictions in cross-validation
aba_plot(model_aba, main = "Basal area")
```

`aba_combine_strata` *Combines a list of ABA models into a single ABA model object*

Description

Combines a list of models (obtained with `aba_build_model`) 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

```
aba_combine_strata(model.list, plotsId = NULL)
```

Arguments

`model.list`	list. stratum-specific models returned by `aba_build_model`
`plotsId`	vector. "plotsId" for ordering row names in the "values" element of the output list

Value

a list with three elements

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

See Also

`aba_build_model` for calibrated ABA model, `aba_plot` for plotting model cross-validation results.

Examples

```
# load Quatre Montagnes dataset
data(quatre_montagnes)
# initialize list of models
model_aba_stratified <- list()
# calibrate basal area prediction model for each stratum
for (i in levels(quatre_montagnes$stratum))
{
  subsample <- which(quatre_montagnes$stratum == i)
  model_aba_stratified[[i]] <-
    aba_build_model(quatre_montagnes[subsample, "G_m2_ha"],
      quatre_montagnes[subsample, 9:76],
      transform = "boxcox", nmax = 4,
      xy = quatre_montagnes[subsample, c("X", "Y")]
    )
}
# combine models in single object
model_aba_stratified <- aba_combine_strata(
  model_aba_stratified,
  quatre_montagnes$plotId
)
# display content of output list
model_aba_stratified$model
model_aba_stratified$stats
summary(model_aba_stratified$values)

# plot field values VS predictions in cross-validation
aba_plot(model_aba_stratified)
```

aba_inference *computes inference from area-based model and predicted values*

Description

computes inference from area-based model and predicted values

Usage

```
aba_inference(
  aba_model,
  r_predictions,
  type = c("SRS", "ED", "D", "STR", "SYNT"),
  r_mask = NULL
)
```

Arguments

aba_model	a model returned by aba_build_model or aba_combine_strata
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 data frame with estimation of parameter value and standard deviation of estimation for all required estimators.

`aba_metrics` *Function for area-based metrics computation*

Description

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

1. for all points: total number `ntot`, percentage of points above minimum height `p_hmin`, percentage of points in height bins `H.propZ1_Z2`,
2. for first return points: percentage above minimum height `p_1st_hmin`,
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

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

.aba_metrics
```

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

Bouvier et al. 2015. Generalizing predictive models of forest inventory attributes using an area-based approach with airborne LiDAR data. Remote Sensing of Environment 156, pp. 322-334. doi:10.1016/j.rse.2014.10.004

See Also

`cloud_metrics`, `stdmetrics`, `clouds_metrics`

Examples

```
# load LAS file
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)

# set number of threads
lidR::set_lidr_threads(2)
# extract two point clouds from LAS object
llas <- lidR::clip_circle(las_chablais3,
                          c(974350, 974390),
                          c(6581680, 6581680), 10)
# normalize point clouds
llas <- lapply(llas, function(x) {
  lidR::normalize_height(x, lidR::tin())
})
# computes metrics
m <- clouds_metrics(llas, ~ aba_metrics(
 Z, Intensity, ReturnNumber, Classification, 2
))
head(m[,1:5])
```

`aba_plot` *Plots observed VS values predicted in leave one out cross validation of an* `aba_build_model`

Description

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

Usage

```
aba_plot(aba_model, disp_text = F, col = NULL, add_legend = NULL, ...)
```

Arguments

`aba_model`	list. as returned by `aba_build_model`
`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
`add_legend`	list. parameters to be passed to `legend`. In case of a stratified model, legend is automatically set up.
`...`	other parameters to be passed to `plot`, `xlab` and `ylab` are automatically added

Value

nothing

Examples

```
# load Quatre Montagnes dataset
data(quatre_montagnes)
# build ABA model for basal area, with three metrics as predictors
model_aba <- aba_build_model(quatre_montagnes$G_m2_ha,
                             quatre_montagnes[, c("zpcum8", "ipcumzq70", "p_hmin")],
  transform = "log", nmax = 2
)

# plot field values VS predictions in cross-validation
aba_plot(model_aba, main = "Basal area")
```

`aba_predict` *Mapping of ABA prediction models*

Description

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

Usage

```
aba_predict(
  model_aba,
  metrics_map,
  stratum = NULL,
  add_error = FALSE,
  pkg = "terra",
  ...
)
```

Arguments

`model_aba`	model returned by `aba_build_model` or `aba_combine_strata`
`metrics_map`	raster. metrics returned e.g by `pixel_metrics`
`stratum`	string. indicates which layer of metrics.map contains the `stratum` in case of a stratified `aba.model`. The layer should have a RAT including a column with the same name (see `is.factor`).
`add_error`	boolean. indicates whether errors sampled from a normal distribution N(0, sigma(residuals)) should be added to fitted values; implemented only for `log` transformation case
`pkg`	raster output format. Use pkg = "terra|raster|stars" to get an output in SpatRaster, RasterLayer or stars format.
`...`	other parameters to be passed to `predict.lm`, e.g. `interval = "prediction"`,

Value

a raster of predictions obtained by applying the model `aba_build_model` to the observations in `metrics_map`

See Also

aba_build_model for model fitting and aba_combine_strata for combining stratified models, clean_raster for applying spatial mask and value thresholds to a raster.

Examples

```
# load data
data(quatre_montagnes)
# build model
model_aba <- aba_build_model(quatre_montagnes$G_m2_ha, quatre_montagnes[, 9:76],
  transform = "boxcox"
)
# build example raster to apply model
quatre_montagnes$X <- rep(1:8, 12)
quatre_montagnes$Y <- rep(1:12, each = 8)
metrics_map <- terra::rast(quatre_montagnes[, c(2, 3, 9:76)], type = "xyz")
predict_map <- aba_predict(model_aba, metrics_map)

# plot map
terra::plot(predict_map, main = "predictions")
```

---

add_vegetation_indices

Add vegetation indices on a IRC image

Description

Computes vegetation indices from the Red, Green and Infra-Red bands of an IRC image and adds them as additional bands or columns. Acronyms are listed on `https://www.nv5geospatialsoftware.com/docs/BroadbandGreenness.html`. If the Blue band is also present, additional indices are computed.

Usage

```
add_vegetation_indices(r, all = FALSE, scale = 255)
```

Arguments

`r`	raster or data.frame. Should contain bands or columns with names nir, r, g (and b)
`all`	boolean. indicates whether all indices should be computed; default:FALSE, only grvi, sr and ndvi are calculated
`scale`	numeric. values in bands are scaled from range [0, scale] to [0, 1]

Value

a raster 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),
b = c(20, 60, 0))
add_vegetation_indices(df, all = TRUE)
```

`boxcox_itr` *Inverse Box-Cox transformation*

Description

Inverse Box-Cox transformation

Usage

```
boxcox_itr(x, lambda)
```

Arguments

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

Value

a vector or raster of transformed values

See Also

`boxcox_tr` Box-Cox transformation, `boxcox_itr_bias_cor` inverse Box-Cox transformation with bias correction.

Examples

```
x <- 1:10
boxcox_itr(x, 0)
boxcox_itr(x, 0.5)
boxcox_itr(x, 2)
boxcox_itr(boxcox_tr(x, 2), 2)

# plot functions
curve(boxcox_itr(x, 0), 0, 3,
  col = "blue", main = "inverse Box Cox transf.",
  xlab = "x", ylab = "inverse Boxcox(x, lambda)"
)
curve(boxcox_itr(x, 1.5), 0, 3, col = "red", add = TRUE)
curve(boxcox_itr(x, 0.5), 0, 3, col = "black", add = TRUE)
curve(boxcox_itr(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
)
```

boxcox_itr_bias_cor *Inverse Box-Cox transformation with bias correction*

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, uses ‘n-p‘ instead of ‘n-1‘, with ‘p‘ the number of variables in the model.

Usage

```
boxcox_itr_bias_cor(x, lambda, varmod)
```

Arguments

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

Value

a vector or raster

References

Xiaojun Pu and Michael Tiefelsdorf, 2015. A variance-stabilizing transformation to mitigate biased variogram estimation in heterogeneous surfaces with clustered samples. doi:10.1007/9783-319227863_24

See Also

boxcox_tr Box-Cox transformation, boxcox_itr inverse Box-Cox transformation.

Examples

```
x <- 1:10
boxcox_itr(x, 0.3)
boxcox_itr_bias_cor(x, 0.3, 0)
boxcox_itr_bias_cor(x, 0.3, 2)

# plot functions
curve(boxcox_itr(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(boxcox_itr_bias_cor(x, 0.3, 1), 0, 3, col = "red", add = TRUE)
curve(boxcox_itr_bias_cor(x, 0.3, 2), 0, 3, col = "black", add = TRUE)
legend("topleft",
  legend = c(
    "residuals variance  = 2",
    "residuals variance  = 1", "residuals variance not accounted for"
  ),
  col = c("black", "red", "blue"), lty = 1
)
```

`boxcox_tr` *Box-Cox Transformation*

Description

Box-Cox Transformation

Usage

```
boxcox_tr(x, lambda)
```

Arguments

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

Value

a vector or raster of transformed values

See Also

`boxcox_itr` inverse Box-Cox transformation, `boxcox_itr_bias_cor` inverse Box-Cox transformation with bias correction.

Examples

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

# plot functions
curve(boxcox_tr(x, 1.5), 1, 5,
  main = "Box Cox transform", xlab = "x",
  ylab = "Boxcox(x, lambda)", col = "red"
)
curve(boxcox_tr(x, -2), 1, 5, col = "green", add = TRUE)
curve(boxcox_tr(x, 0), 1, 5, col = "blue", add = TRUE)
curve(boxcox_tr(x, 0.5), 1, 5, col = "black", add = TRUE)
curve(boxcox_tr(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
)
```

---

chm_chablais3 *Canopy height model (Chablais 3 plot)*

Description

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

Usage

```
data(chm_chablais3)
```

Format

A PackedSpatRaster object

References

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. pp. 21-22 & 34 https://theses.hal.science/tel-00652698/document

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)
terra::plot(chm_chablais3)
```

cimg2Raster *Cimg to SpatRaster conversion*

Description

converts a cimg object to a SpatRaster object

Usage

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

Arguments

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

Value

A SpatRaster object

See Also

raster2Cimg

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# convert raster to cimg object
chm_cim <- raster2Cimg(chm_chablais3)

# apply filtering
chm_cim_filt <- dem_filtering(chm_cim,
  nl_filter = "Closing",
  nl_size = 3,
  sigma = 0
)$non_linear_image

# convert to SpatRaster
chm_filt <- cimg2Raster(chm_cim_filt, chm_chablais3)

# plot SpatRaster
terra::plot(chm_chablais3)

# plot cimg object
plot(chm_cim)
```

```
# plot filtered cimg object
plot(chm_cim_filt)

# plot filtered SpatRaster
terra::plot(chm_filt)
```

---

circle2Raster *Raster corresponding to circle extent*

Description

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

Usage

```
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
`...`	other parameters to pass to `rast` (e.g. crs)

Value

A SpatRaster object

Examples

```
circle2Raster(100, 100, 20, 1, 5)
```

clean_raster *Applies thresholds and mask to a raster 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

```
clean_raster(r, minmax = c(-Inf, +Inf), mask = NULL)
```

Arguments

`r`	raster object. RasterLayer and SpatRaster are supported.
`minmax`	vector of two numeric values. minimum and maximum thresholds to apply to ‘r‘ values
`mask`	raster object. mask to be applied (multiplication with input raster ‘r‘)

Value

a raster object

Examples

```
# load data
data(quatre_montagnes)
# build model
model_aba <- aba_build_model(quatre_montagnes$G_m2_ha, quatre_montagnes[, 9:76],
  transform = "boxcox"
)
# build example raster to apply model
quatre_montagnes$X <- rep(1:8, 12)
quatre_montagnes$Y <- rep(1:12, each = 8)
metrics_map <- terra::rast(quatre_montagnes[, c(2, 3, 9:76)], type = "xyz")
predict_map <- aba_predict(model_aba, metrics_map)
# create raster mask
mask <- predict_map
# set values to 1 or NA
terra::values(mask) <- rep(c(1, 1, NA), each = 32)
# apply thresholds and mask
predict_map_clean <- clean_raster(predict_map, c(40, 70), mask)

# plot maps
terra::plot(predict_map, main = "Predictions")
terra::plot(mask, main = "Mask", legend = FALSE)
terra::plot(predict_map_clean, main = "Cleaned predictions")
```

clouds_metrics *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

```
clouds_metrics(
  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, aba_metrics, pixel_metrics

Examples

```
# load LAS file
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)

# set number of threads
lidR::set_lidr_threads(2)
# extract four point clouds from LAS object
llas <- list()
llas[["A"]] <- lidR::clip_circle(las_chablais3, 974350, 6581680, 10)
llas[["B"]] <- lidR::clip_circle(las_chablais3, 974390, 6581680, 10)
llas[["C"]] <- lidR::clip_circle(las_chablais3, 974350, 6581640, 10)
# normalize point clouds
llas <- lapply(llas, function(x) {
  lidR::normalize_height(x, lidR::tin())
})

# compute metrics
clouds_metrics(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
  dummy <- which(z >= hmin & rn == 1)
  return(list(
    mean.z = mean(z[dummy]),
    sd.z = stats::sd(z[z > hmin])
  ))
}
clouds_metrics(llas, func = ~ user_func(Z, ReturnNumber, 10))
```

`clouds_tree_metrics` *Computes metrics on trees detected in list of point clouds.*

Description

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

Usage

```
clouds_tree_metrics(llasn, XY, plot_radius, res = 0.5, roi = NULL, func, ...)
```

Arguments

`llasn`	list of `LAS` objects
`XY`	a data frame 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
`roi`	sf object. polygons corresponding to plot shapes. Only trees which apices are inside the polygons are retained for subsequent computations. However, plot surface is still computed as pi * plot_radius^2
`func`	a function to be applied to the attributes of extracted trees (return from internal call to `tree_extraction` function) to compute plot level metrics
`...`	other parameters to be passed to `tree_segmentation`

Details

- calls `tree_segmentation` to segment trees and then `tree_extraction` to extract their features
- computes ‘TreeCanopy_cover_in_plot‘ (proportion of surface of disk of interest which is covered by segmented trees), ‘TreeCanopy_meanH_in_plot‘ (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 (default is `std_tree_metrics`)

Value

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

See Also

tree_segmentation, tree_extraction, std_tree_metrics

Examples

```
# load LAS file
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)

# set number of threads
lidR::set_lidr_threads(2)
# extract two point clouds from LAS object
llas <- lidR::clip_circle(las_chablais3,
                          c(974350, 974390),
                          c(6581680, 6581680), 10)
# normalize point clouds
llas <- lapply(llas, function(x) {
  lidR::normalize_height(x, lidR::tin())
})

# compute metrics with user-defined function
# number of detected trees between 20 and 30 meters and their mean height
# restricted to disks of radius 8 m.
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))
}
XY <- data.frame(X = c(974350, 974390),
                 Y = c(6581680, 6581680))
clouds_tree_metrics(llas,
                    XY,
                    8,
                    res = 0.5,
                    func = user_func
)
#
# same result using a user-input circular roi
roi <- sf::st_as_sf(XY,
                    coords = c("X", "Y"),
                    crs = sf::st_crs(2154)
)
roi <- sf::st_buffer(roi, 8)
clouds_tree_metrics(llas,
                    XY,
                    8,
                    roi = roi,
                    res = 0.5,
                    func = user_func
)
```

`convert_raster` *Raster format conversion*

Description

Function to convert between raster formats. Use pkg = "terra|raster|stars" to get an output in SpatRaster, RasterLayer or stars format. Default is getOption("lidR.raster.default").

Usage

```
convert_raster(r, pkg = NULL)
```

Arguments

r	raster object or file name.
pkg	package name. Use pkg = "terra| stars" to get an output in SpatRaster or stars format

Value

A raster object in the specified format

Examples

```
# load SpatRaster
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)
# convert only if packages stars and raster are installed
# if (require("stars"))
# {
# to stars
# chm_stars <- convert_raster(chm_chablais3, pkg = "stars")
# chm_stars
# convert_raster(chm_stars, pkg = "terra")
# }
```

coregistration — *Tree inventory and canopy height model coregistration*

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, 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 SpatRaster returned by `rasters_moving_cor`, second a data.frame returned by `raster_local_max`

References

Monnet, J.-M. and Mermin, E. 2014. Cross-Correlation of Diameter Measures for the Co-Registration of Forest Inventory Plots with Airborne Laser Scanning Data. Forests 2014, 5(9), 2307-2326, doi:10.3390/f5092307

See Also

`rasters_moving_cor`, `raster_local_max`

Examples

```
# 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 <- raster_xy_mask(rbind(c(20, 20), c(20, 20)), c(9, 9), tree_mask,
binary = TRUE)

# simulate chm raster
chm <- terra::rast(extent = c(0, 40, 0, 40), resolution = 1, crs = NA)
xy <- terra::xyFromCell(chm, 1:(ncol(chm) * nrow(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 <- terra::rast(cbind(xy, pmax(z1, z2, z3)), type = "xyz") #+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
terra::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"
)
```

`create_disk` *Disk-shaped matrix mask*

Description

Creates a matrix with TRUE values shaping a centered disk

Usage

```
create_disk(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

```
create_disk(7)
```

`dem_filtering` *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

```
dem_filtering(
  dem,
  nl_filter = "Closing",
  nl_size = 5,
  sigma = 0.3,
  padding = TRUE,
  sigmap = NULL
)
```

Arguments

`dem`	cimg object (e.g. obtained with `as.cimg`) or SpatRaster object (e.g. obtained with `rast`)
`nl_filter`	string. type of non-linear filter to apply: "None", "Closing" or "Median"
`nl_size`	numeric. kernel width in pixels for non-linear filtering
`sigma`	numeric or matrix. If a single number is provided, `sigma` is the standard deviation of the Gaussian filter, 0 corresponds to no smoothing. Unit is pixel in case `dem` is a cimg object, SpatRaster units otherwise. In case of a 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
`sigmap`	deprecated (numeric or matrix). (old name for `sigma` parameter, retained for backward compatibility, overwrites `sigma` if provided, unit is pixel whatever the class of `dem`)

Value

A list of two cimg objects or a SpatRaster object with image after non-linear filter and image after both filters

See Also

`maxima_detection`, filters of imager package: `mclosing`, `medianblur`, `deriche`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# filtering with median and Gaussian smoothing
im <- dem_filtering(chm_chablais3, nl_filter = "Median", nl_size = 3, sigma = 0.8)

# filtering with median filter and value-dependent Gaussian smoothing
# (less smoothing for values between 0 and 15)
im2 <- dem_filtering(chm_chablais3,
  nl_filter = "Median", nl_size = 3,
  sigma = cbind(c(0.2, 0.8), c(0, 15))
)

# plot original image
terra::plot(chm_chablais3, main = "Initial image")

# plot image after median filter
terra::plot(im$non_linear_image, main = "Median filter")

# plot image after median and Gaussian filters
terra::plot(im$smoothed_image, main = "Smoothed image")

# plot image after median and value-dependent Gaussian filters
terra::plot(im2$smoothed_image, main = "Value-dependent smoothing")
```

`edge_detection` *Edge detection in gap image*

Description

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

Usage

```
edge_detection(gaps, inside = TRUE)
```

Arguments

`gaps`	SpatRaster 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 SpatRaster object where edges are labelled as 1.

See Also

`gap_detection`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# fill NA values in canopy height model
chm_chablais3[is.na(chm_chablais3)] <- 0

# gap detection with distance larger than canopy height / 2
gaps <- gap_detection(chm_chablais3,
  ratio = 2, gap_max_height = 1, min_gap_surface = 10,
  gap_reconstruct = TRUE
)

# edge detection
edges_inside <- edge_detection(!is.na(gaps$gap_id))
edges_outside <- edge_detection(!is.na(gaps$gap_id), inside = FALSE)

# edge proportion
sum(terra::values(edges_inside)) / (nrow(edges_inside) * ncol(edges_inside))
sum(terra::values(edges_outside)) / (nrow(edges_outside) * ncol(edges_outside))

# plot original image
terra::plot(chm_chablais3, main = "Initial image")

# plot binary image of gaps
terra::plot(gaps$gap_id > 0, main = "Gaps", col = "green", legend = FALSE)

# plot edges
terra::plot(edges_inside, main = "Edges (inside)", legend = FALSE)
terra::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 extention 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

Value

a list of data.frame containing the coordinates of polygons

See Also

`pointList2poly`

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")
```

gap_detection *Gap detection in a Canopy Height Model*

Description

Performs gaps detection on a canopy height model provided as object of class SpatRaster-class, or computed from the point cloud of objects of class LAS-class or LAScatalog-class. Function dem_filtering 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

```
gap_detection(
  las,
  res = 1,
  ratio = 2,
  gap_max_height = 1,
  min_gap_surface = 25,
  max_gap_surface = +Inf,
  closing_height_bin = 1,
  nl_filter = "Median",
  nl_size = 3,
  gap_reconstruct = FALSE
)
```

Arguments

`las`	An object of class SpatRaster-class, LAS-class or LAScatalog-class
`res`	numeric. The size of a grid cell in point cloud coordinates units, used to rasterize the point cloud. In case the `las` argument is a `SpatRaster` `res` is not used.
`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 between canopy height and gap distance
`nl_filter`	string. type of non-linear filter to apply to canopy height model to remove artefacts, should be an option of `dem_filtering`
`nl_size`	numeric. kernel width in pixel for non-linear filtering
`gap_reconstruct`	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 `SpatRaster` object with three layers: gap labels, gap surface and canopy height model after filter.

See Also

`dem_filtering`, `edge_detection`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# fill NA values in canopy height model
chm_chablais3[is.na(chm_chablais3)] <- 0

# gap detection with distance larger than canopy height / 2
gaps <- gap_detection(chm_chablais3, 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 <- gap_detection(chm_chablais3,
  ratio = 2, gap_max_height = 1, min_gap_surface = 0,
  gap_reconstruct = TRUE
)

# gap detection without distance criterion
gaps2 <- gap_detection(chm_chablais3, ratio = NULL, gap_max_height = 1,
min_gap_surface = 0)

# gap id and corresponding surface for third detection parameters
table(terra::values(gaps2$gap_id)) * terra::res(gaps2$gap_id)[1]^2

# plot original image
terra::plot(chm_chablais3, main = "Initial image")

# plot binary image of gaps
terra::plot(gaps$gap_id > 0, main = "Gaps", col = "green", legend = FALSE)
```

```
terra::plot(gaps1$gap_id > 0, main = "Gaps, with reconstruction", col = "green", legend = FALSE)
terra::plot(gaps2$gap_id > 0, main = "Gaps, no width criterion", col = "green", legend = FALSE)

# plot filtered CHM
terra::plot(gaps2$filled_chm, main = "Filtered CHM")
```

---

height_regression *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

```
height_regression(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 `tree_matching`
`plot`	boolean. indicates whether results should be plotted
`species`	vector of strings. species for standardized color use by call to `species_color`
`...`	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

`tree_matching`

Examples

```
# 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 <- tree_matching(ref_trees, def_trees)

# height regression
reg <- height_regression(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
```

hist_detection *Histogram of detection*

Description

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

Usage

```
hist_detection(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 `tree_matching`
`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

`tree_matching`

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 <- tree_matching(ref_trees, def_trees)
match2 <- tree_matching(ref_trees, def_trees, delta_ground = 2, h_prec = 0)
#
# corresponding number of detections
hist_detection(ref_trees, def_trees, match1)
hist_detection(ref_trees, def_trees, match2)
```

`hist_stack` *Stacked histogram*

Description

Stacked histogram

Usage

```
hist_stack(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

`las_chablais3` *las data in France (Chablais 3 plot)*

Description

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

Format

A compressed LAS file

Details

Additional information about the data

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

Source

Monnet J.-M. INRAE

References

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. pp. 21-22. `https://theses.hal.science/tel-00652698/document`

See Also

`chm_chablais3`, `tree_inventory_chablais3`

Examples

```
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)
las_chablais3
```

`lma_check` *Checks linear model assumptions of a multiple regression model*

Description

The performed tests are:

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

Usage

```
lma_check(formule, df, max.pvalue = 0.05, max.vif = 5)
```

Arguments

`formule`	formula. model to be evaluated
`df`	data.frame. data to evaluate the model
`max.pvalue`	numeric. maximum p-value of variables included in the model
`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 < `max.pvalue`
- a boolean: are all tests implemented by `gvlma` false
- a boolean: is the variance inflation factor computed with `vif` of all variables < `max.vif`

Examples

```
# load Quatre Montagnes dataset
data(quatre_montagnes)
# fit lm model
model <- lm(G_m2_ha ~ zmax + zq95, data = quatre_montagnes)
lma_check(eval(model$call[[2]]), quatre_montagnes)
# trying with Box-Cox transformation of dependent variable
# and other independent variables
model <- lm(boxcox_tr(G_m2_ha, -0.14) ~ Tree_meanH + Tree_density + zpcum7, data = quatre_montagnes)
lma_check(eval(model$call[[2]]), quatre_montagnes)
```

`maxima_detection` *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

```
maxima_detection(dem, dem.res = 1, max.width = 11, jitter = TRUE)
```

Arguments

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

Value

A cimg object / SpatRaster object which values correspond to the radius (n) in pixels / meters of the square window (width 2n+1) where the center pixel is global maximum (tested up to the `max.width` parameter)

See Also

`dem_filtering`, `maxima_selection`, `tree_segmentation`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# maxima detection
maxi <- maxima_detection(chm_chablais3)

# plot original image
terra::plot(chm_chablais3, main = "Initial image")

# plot maxima image
terra::plot(maxi, main = "Local maxima")
```

`maxima_selection` *Image maxima selection based on values and neighborhood of local maxima*

Description

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

1. values 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.

Make sure that the `max.width` parameter in `maxima_detection` is consistent with the selection parameters (e.g. do not select with `dmin = 7` if values were only tested up to `max.width` the default value which is approx. 5.5 m).

Usage

```
maxima_selection(maxi, dem_nl, hmin = 5, dmin = 0, dprop = 0.05)
```

Arguments

`maxi`	cimg object or SpatRaster object. image with local maxima (typically output from `maxima_detection`, image values correspond to neighborhood radius on which pixels are global maxima in the initial image)
`dem_nl`	cimg object or SpatRaster 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 SpatRaster 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

maxima_detection, tree_segmentation

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# maxima detection
maxi <- maxima_detection(chm_chablais3)

# several maxima selection settings
selected_maxi_hmin <- maxima_selection(maxi, chm_chablais3, hmin = 15)
selected_maxi_dm <- maxima_selection(maxi, chm_chablais3, dmin = 2.5)
selected_maxi <- maxima_selection(maxi, chm_chablais3, dmin = 1, dprop = 0.1)

# corresponding count number of remaining maxima
table(terra::values(maxi))
table(terra::values(selected_maxi_hmin))
table(terra::values(selected_maxi_dm))
table(terra::values(selected_maxi))

# plot original image
terra::plot(chm_chablais3, main = "Initial image")

# plot maxima images, original and first case
terra::plot(maxi, main = "Local maxima")
terra::plot(selected_maxi, main = "Selected maxima")
```

---

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

Description

Plot of matched pairs of detected and reference trees

Usage

```
plot_matched(lr, ld, matched, chm = NULL, plot_border = 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 `tree_matching`
`chm`	raster object. raster for background display
`plot_border`	sf or SpatVector object. plot boundaries for display
`...`	Additional arguments to be used by `plot`

Value

no return

See Also

`tree_matching`, `hist_detection`

Examples

```
# 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 <- tree_matching(ref_trees, def_trees)
match2 <- tree_matching(ref_trees, def_trees, delta_ground = 2, h_prec = 0)

# 2D display of matching results
plot_matched(ref_trees, def_trees, match1, xlab = "X", ylab = "Y")
plot_matched(ref_trees, def_trees, match2, xlab = "X", ylab = "Y")
```

`plot_tree_inventory` *Displays a map of tree inventory data*

Description

displays tree inventory data

Usage

```
plot_tree_inventory(xy, height = NULL, diam = NULL, species = NULL, ...)
```

Arguments

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

Value

no return

See Also

`species_color` for a table of species and associated colors

Examples

```
# load tree inventory data from plot Chablais 3
data("tree_inventory_chablais3")

# display tree inventory
plot_tree_inventory(tree_inventory_chablais3[, c("x", "y")],
  diam = tree_inventory_chablais3$d, col = "red",
  pch = tree_inventory_chablais3$e,
  xlab = "X", ylab = "Y"
)

# display tree inventory with CHM background
data("chm_chablais3")
chm_chablais3 <- terra::rast(chm_chablais3)
terra::plot(chm_chablais3, col = gray(seq(0, 1, 1 / 255)))
plot_tree_inventory(tree_inventory_chablais3[, c("x", "y")],
  height = tree_inventory_chablais3$h,
  species = tree_inventory_chablais3$s,
  add = TRUE
)
```

`pointList2poly` *Convert a list of points into spatial polygons object*

Description

Converts a list of points specifying polygons into a spatial object

Usage

```
pointList2poly(points_list, df = NULL, ...)
```

Arguments

`points_list`	list of data frames of xy coordinates. In each data.frame the last row must be the same as the first row
`df`	data.frame. Optional data.frame to be associated to polygons
`...`	arguments to be passed to `st_sfc`

Value

a simple feature collection with POLYGON geometry.

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 sf object
ellipses1 <- pointList2poly(ellipses)
ellipses1
# Convert to sf object with user-defined data.frame
ellipses2 <- pointList2poly(ellipses, df = data.frame(info = 1:2))

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

`points2DSM` *Digital Surface Model*

Description

Creates a Digital Surface Model from a LAS object. From version 4.0.0 relies on `rasterize_canopy`. Maintained for backward compatibility but a direct call to this function should be preferred. 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

`.las`	`LAS` object or XYZ matrix/data.frame
`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 SpatRaster object.

See Also

`points2DTM` for Digital Terrain Model computation.

Examples

```
# load LAS file
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)

# set number of threads
lidR::set_lidr_threads(2)
# create a digital surface model with first-return points, resolution 0.5 m
dsm <- points2DSM(lidR::filter_first(las_chablais3), res = 0.5)

# display raster
terra::plot(dsm)
```

`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 `rasterize_terrain` with algorithm `tin`. In case a LAS object is provided, only points classified as ground or water (2 or 9) will be used.

Usage

```
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 SpatRaster object

See Also

points2DSM for Digital Surface Model computation.

Examples

```
# load LAS file
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)

# set number of threads
lidR::set_lidr_threads(2)
# create digital terrain model with points classified as ground
dtm <- points2DTM(las_chablais3)

# display raster
terra::plot(dtm)
```

---

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

```
polar2Projected(
  x,
  y,
  z = 0,
  azimuth,
  dist,
  slope = 0,
  declination = 0,
  convergence = 0,
  diameter = 0
)
```

Arguments

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

Value

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

See Also

`plot_tree_inventory` 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
)
```

`quatre_montagnes` *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(quatre_montagnes)
```

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
7. `D_mean_cm` mean tree diameter at breast height (1.3 m) in cm
8. `stratum` forest ownership (public or private)
9. `[, 9:60]` point cloud metrics computed from ALS, see `aba_metrics`
10. `[, 61:73]` metrics derived from tree segmentation in ALS data, see `std_tree_metrics`
11. `[, 74:76]` terrain statistics, see `terrain_points_metrics`

References

Monnet, J.-M. 2021. Tutorial on modeling forest parameters with ALS data. ABA data preparation https://lidar.pages.mia.inra.fr/lidaRtRee/articles/area-based.1.data.preparation.html

Examples

```
data(quatre_montagnes)
summary(quatre_montagnes)
```

raster2Cimg *SpatRaster to Cimg conversion*

Description

converts a SpatRaster object to cimg object. NA values in raster are replaced.

Usage

```
raster2Cimg(r, NA_replace = 0, maxpixels = 1e+10)
```

Arguments

`r`	SpatRaster object. raster of canopy height model, preferably filtered to avoid effect of holes on volume and surface computation
`NA_replace`	numeric. value to replace NA values with.
`maxpixels`	numeric. maximum number of pixels to be converted to cimg (argument passed to `as.cimg`).

Value

A cimg object

See Also

`cimg2Raster`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

chm_cim <- raster2Cimg(chm_chablais3)
chm_cim
summary(chm_cim)

# plot SpatRaster
terra::plot(chm_chablais3)

# plot cimg object
plot(chm_cim)
```

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

raster_b	SpatRaster. raster to correlate with largest extent
raster_s	SpatRaster. raster to correlate with smallest extent
mask	SpatRaster. mask of area to correlate
small.SC	boolean. is the small raster already standardized and centered ?

Value

A numeric

See Also

rasters_moving_cor to compute correlation between rasters for different translations

Examples

```
# create raster
r_b <- terra::rast(xmin = 0, xmax = 40, ymin =0 , ymax = 40,
resolution = 1, crs = NA)
xy <- terra::xyFromCell(r_b, 1:(nrow(r_b) * ncol(r_b)))

# add Gaussian surface and noise
z <- 3 * exp(-((xy[, 1] - 20)^2 + (xy[, 2] - 20)^2 / 2) / 6)
r_b <- terra::rast(cbind(xy, z), type = "xyz")

# create circular mask of radius 5
z_mask <- (xy[, 1] - 20)^2 + (xy[, 2] - 20)^2 < 5^2
r_mask <- terra::rast(cbind(xy, z_mask), type = "xyz")

# create small raster of size 20
r_s <- terra::crop(r_b, terra::ext(c(10, 30, 10, 30)))

# add noise to small raster
terra::values(r_s) <- terra::values(r_s) + rnorm(ncol(r_s) * nrow(r_s), 0, 0.5)
r_mask <- terra::crop(r_mask, terra::ext(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
terra::plot(r_b, main = "Large raster")
# display small raster
terra::plot(r_s, main = "Small raster")
# display mask
terra::plot(r_mask, main = "Computation mask")
```

rasters_moving_cor *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

```
rasters_moving_cor(raster_b, raster_s, mask = NULL, buffer = 19, step = 0.5)
```

Arguments

`raster_b`	SpatRaster. raster to correlate with largest extent
`raster_s`	SpatRaster. raster to correlate with smallest extent
`mask`	SpatRaster. 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 SpatRaster. 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

`raster_local_max` to extract local maximum of resulting correlation raster, `rasters2Cor`

Examples

```
# create raster
r_b <- terra::rast(xmin = 0, xmax = 40, ymin =0 , ymax = 40,
resolution = 1, crs = NA)
xy <- terra::xyFromCell(r_b, 1:(nrow(r_b) * ncol(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 <- terra::rast(cbind(xy, z1 + z2 + z3), type = "xyz")

# create small raster
r_s <- terra::crop(r_b, terra::ext(c(15, 25, 15, 25)))
# offset raster by (-2, -2)
terra::ext(r_s) <- c(13, 23, 13, 23)

# compute correlations for translations inside buffer
rr <- rasters_moving_cor(r_b, r_s, buffer = 6, step = 1)
rr

# display large raster
terra::plot(r_b, main = "Large raster")
# display small raster
terra::plot(r_s, main = "Small raster")
# display correlation
terra::plot(rr,
  xlab = "X translation", ylab = "Y translation",
  main = "Correlation between rasters"
)
```

`raster_chull_mask` *Raster mask of convex hull*

Description

creates raster mask corresponding to the convex hull of xy positions

Usage

```
raster_chull_mask(xy, r)
```

Arguments

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

Value

a SpatRaster with 0 or 1

See Also

raster_xy_mask

Examples

```
# create raster
r <- terra::rast(extent = c(0, 40, 0, 40), resolution = 1, crs = "epsg:2154")


# xy positions
xy <- data.frame(
  x = c(10, 20, 31.25, 15),
  y = c(10, 20, 31.25, 25)
)
# compute mask
mask1 <- raster_chull_mask(xy, r)

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

`raster_local_max` *Statistics of raster local maximum*

Description

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

Usage

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

Arguments

`r`	SpatRaster. typically output of `rasters_moving_cor`
`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

rasters_moving_cor, coregistration for application to the coregistration of tree inventory data with canopy height models

Examples

```
# create raster
r_b <- terra::rast(xmin = 0, xmax = 40, ymin =0 , ymax = 40,
resolution = 1, crs = NA)
xy <- terra::xyFromCell(r_b, 1:(nrow(r_b) * ncol(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 <- terra::rast(cbind(xy, z1 + z2 + z3), type = "xyz")

# create small raster
r_s <- terra::crop(r_b, terra::ext(c(15, 25, 15, 25)))
# offset raster by (-2, -2)
terra::ext(r_s) <- c(13, 23, 13, 23)

rr <- rasters_moving_cor(r_b, r_s, buffer = 6, step = 1)
loc_max <- raster_local_max(rr)
loc_max

# plot raster
terra::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
)
```

---

raster_metrics

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

---

Description

Computes 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 with the XY coordinates of the larger cells, or as SpatRaster.

Usage

```
raster_metrics(
  r,
  res = 20,
  fun = function(x) {
     data.frame(mean = mean(x[, 3]), sd = stats::sd(x[, 3]))
 },
  start = c(0, 0),
  output = "raster"
)
```

Arguments

`r`	SpatRaster object, data.frame with xy coordinates in two first columns, or POINT `sf` spatial object
`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$layer to access raster values) / data.frame (use x$colum_name to access values)
`start`	vector of x and y coordinates for the reference raster. Default is (0,0) meaning that the grid aligns on (0,0).
`output`	string. indicates the class of output object "raster" for a SpatRaster 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 SpatRaster object

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)
# replace NA with zeros
chm_chablais3[is.na(chm_chablais3)] <- 0

# raster metrics from raster
metrics1 <- raster_metrics(chm_chablais3, 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 <- raster_metrics(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
terra::plot(metrics1)
# display data.frame metrics
terra::plot(terra::rast(metrics2, type = "xyz"))
```

---

raster_xy_mask — *Raster mask by union of buffers around xy positions*

Description

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

Usage

```
raster_xy_mask(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

raster_chull_mask

Examples

```
# create raster
r <- terra::rast(xmin=0, xmax = 40, ymin = 0, ymax = 40, resolution = 1, crs= NA )

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

```
# compute mask
mask1 <- raster_xy_mask(xy, c(5, 8, 5, 5), r)
mask2 <- raster_xy_mask(xy, c(5, 8, 5, 5), r, binary = FALSE)

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

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

`raster_zonal_stats` *Image statistic in segment*

Description

compute zonal statistic of an image

Usage

```
raster_zonal_stats(segms, dem_nl, fun = max)
```

Arguments

`segms`	cimg or SpatRaster object. image with segments id (e.g. from `segmentation`)
`dem_nl`	cimg or SpatRaster object. image to compute statistic from
`fun`	function to compute statistics from values in each segment

Value

A cimg object or raster object with values of the statistic

See Also

`segmentation`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# median filter
chm_chablais3 <- dem_filtering(chm_chablais3,
  nl_filter = "Median", nl_size = 3,
  sigma = 0
)$non_linear_image

# maxima detection
```

```
maxi <- maxima_detection(chm_chablais3)

# segmentation
seg_maxi <- segmentation(maxi, chm_chablais3)

# compute image of maximum value in each segment
max_in_segment <- raster_zonal_stats(seg_maxi, chm_chablais3)

# plot original image
terra::plot(chm_chablais3, main = "Median filter")

# plot segments and image of max value inside segments
seg_maxi[seg_maxi == 0] <- NA
terra::plot(seg_maxi %% 8, main = "Segments", col = rainbow(8))
terra::plot(max_in_segment, main = "Max value in segment")
```

`segmentation` *Image segmentation by seed-based watershed algorithm*

Description

performs a seed-based watershed segmentation (wrapper for `watershed`)

Usage

```
segmentation(maxi, dem_nl)
```

Arguments

`maxi`	cimg or SpatRaster object. image with seed points (e.g. from `maxima_detection` or `maxima_selection`)
`dem_nl`	cimg or SpatRaster object. image for seed propagation (typically initial image used for maxima detection).

Value

A cimg object or SpatRaster object with segments id

See Also

`maxima_detection`, `maxima_selection`, `seg_adjust`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# median filter
chm_chablais3 <- dem_filtering(chm_chablais3,
  nl_filter = "Median", nl_size = 3,
  sigma = 0
)$non_linear_image

# maxima detection
maxi <- maxima_detection(chm_chablais3)

# maxima selection
selected_maxi <- maxima_selection(maxi, chm_chablais3, dm = 1, dprop = 0.1)

# segmentation
seg_maxi <- segmentation(maxi, chm_chablais3)
seg_selected_maxi <- segmentation(selected_maxi, chm_chablais3)

# plot original image
terra::plot(chm_chablais3, main = "Median filter")

# plot segmented image
# replace segment with id 0 (not a tree) with NA
seg_maxi[seg_maxi == 0] <- NA
terra::plot(seg_maxi %% 8, main = "Segments, no maxima selection",
col = rainbow(8))
seg_selected_maxi [seg_selected_maxi == 0] <- NA
terra::plot(seg_selected_maxi %% 8, main = "Segments, maxima selection",
col = rainbow(8))
```

seg_adjust *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

```
seg_adjust(dem_w, dem_wh, dem_nl, prop = 0.3, min.value = 2, min.maxvalue = 5)
```

Arguments

`dem_w`	cimg or SpatRaster object. image with segments id, without 0 values
`dem_wh`	cimg or SpatRaster object. image with max value inside segment
`dem_nl`	cimg or SpatRaster object. image with initial values
`prop`	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`	numeric. threshold for removel of pixels which initial values are lower (`dem_nl` < `min.value`)
`min.maxvalue`	numeric. threshold for complete removal of segments which maximum value height is smaller to the threshold (`dem_wh` < `min.maxvalue`)

Value

A cimg or SpatRaster object: image with modified segments.

See Also

maxima_detection, maxima_selection

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# median filter
chm_chablais3 <- dem_filtering(chm_chablais3,
  nl_filter = "Median", nl_size = 3,
  sigma = 0
)$non_linear_image

# maxima detection and selection
maxi <- maxima_detection(chm_chablais3)
selected_maxi <- maxima_selection(maxi, chm_chablais3, dm = 1, dprop = 0.1)

# segmentation
seg_selected_maxi <- segmentation(selected_maxi, chm_chablais3)

# max value in segments
max_in_segment <- raster_zonal_stats(seg_selected_maxi , chm_chablais3)

# segmentation modification
seg_modif1 <- seg_adjust(seg_selected_maxi , max_in_segment,
  chm_chablais3,
  prop = 0.5
)
seg_modif2 <- seg_adjust(seg_selected_maxi , max_in_segment,
  chm_chablais3,
  prop = 0, min.value = 5, min.maxvalue = 10
)

# plot initial segmented image
# seg_selected_maxi[seg_selected_maxi == 0] <- NA
terra::plot(seg_selected_maxi %% 8, main = "Initial segments", col = rainbow(8))
# seg_modif1[seg_modif1 == 0] <- NA
terra::plot(seg_modif1 %% 8, main = "Modified segments 1", col = rainbow(8))
seg_modif2[seg_modif2 == 0] <- NA
terra::plot(seg_modif2 %% 8, main = "Modified segments 2", col = rainbow(8))
```

---

species_color *Table of species names, abreviations and display colors*

Description

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

Usage

```
species_color()
```

Value

A data frame with species name, color, coniferous (C) / broadleaf (B) type, and name abreviation GESP of GEnus and SPecies

See Also

`plot_tree_inventory` for tree inventory display

Examples

```
# load table
tab.species <- species_color()
head(tab.species)
summary(tab.species)
```

`std_tree_metrics` *Computation of tree metrics*

Description

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

Usage

```
std_tree_metrics(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 `tree_extraction`. `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

`tree_extraction`, `clouds_tree_metrics`, `raster_metrics`

Examples

```
# 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)
std_tree_metrics(trees, area_ha = 0.1)
```

`terrain_points_metrics` *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 (`rasterize_terrain` with `tin` algorithm), otherwise it is the mean of the points altitude range.

Usage

```
terrain_points_metrics(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.
`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

```
# set number of threads
lidR::set_lidr_threads(2)
# 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
terrain_points_metrics(XYZ)
terrain_points_metrics(XYZ, centre = c(5, 5), r = 5)
# with a LAS object
LASfile <- system.file("extdata", "las_chablais3.laz", package="lidaRtRee")
las_chablais3 <- lidR::readLAS(LASfile)
terrain_points <- lidR::filter_ground(las_chablais3)
terrain_points_metrics(terrain_points)
terrain_points_metrics(terrain_points, centre = c(974360, 6581650), r = 10)
```

`tree_detection` *Tree detection*

Description

Performs tree detection by applying the functions `tree_segmentation` and `tree_extraction` to objects of class `SpatRaster-class`, `LAS-class` or `LAScatalog-class`

Usage

```
tree_detection(las, res = 1, ROI = NULL, normalize = FALSE, crown = FALSE, ...)
```

Arguments

`las`	An object of class `SpatRaster-class`, `LAS-class` or `LAScatalog-class`
`res`	numeric. The size of a grid cell in point cloud coordinates units, used to rasterize the point cloud. In case the `las` argument is a `SpatRaster` `res` is not used.
`ROI`	spatial polygons in sf/sfc format, in the same CRS as argument `las`. geometric object that defines the region where tree detection has to be performed. In case the input is of class `LAScatalog-class`, the chunk buffer set with `engine_options` is applied to the point cloud to prevent border effects, but only treetops lying within the `ROI` are returned.
`normalize`	boolean. Should the point cloud be normalized before detection (not applicable if `las` argument is a `SpatRaster`) ?
`crown`	Parameter passed to `tree_extraction`
`...`	Parameters passed to `tree_segmentation`

Value

A sf collection of POINTs with 7 fields: tree id, local maximum stats (height, dominance radius), segment stats (surface and volume), coordinates (x and y). In case argument `crown` is `TRUE`, a `crown` field containing the WKT geometry of the 2D crown is also present.

References

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. Section 6.2 https://theses.hal.science/tel-00652698/document

Monnet, J.-M., Mermin, E., Chanussot, J., Berger, F. 2010. Tree top detection using local maxima filtering: a parameter sensitivity analysis. Silvilaser 2010, the 10th International Conference on LiDAR Applications for Assessing Forest Ecosystems, September 14-17, Freiburg, Germany, 9 p. https://hal.science/hal-00523245/document

See Also

`tree_segmentation`, `tree_extraction`

Examples

```
# load canopy height model
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)
# create polygon of region of interest
ROI <- sf::st_polygon(list(cbind(
 c(974360, 974360, 974380, 974380, 974360),
 c(6581640, 6581680, 6581680, 6581640, 6581640)
)))
# convert to sfc and set projection
ROI = sf::st_sfc(ROI)
sf::st_crs(ROI) <- terra::crs(chm_chablais3)
```

```
#
# tree detection
trees <- tree_detection(chm_chablais3)
# plot results
# canopy height model background
terra::plot(chm_chablais3)
# detected trees
plot(trees["h"], add = TRUE, cex = trees$h/20, col = "black")
#
# tree detection in ROI and minimum tree height set to 10
trees_ROI <- tree_detection(chm_chablais3, ROI = ROI, hmin = 10, crown = TRUE)
# create polygons from WKT field
trees_ROI_crowns <- sf::st_as_sf(sf::st_drop_geometry(trees_ROI), wkt = "crown")
# plot results
# canopy height model background
terra::plot(chm_chablais3)
# detected trees
plot(trees_ROI["h"], add = TRUE, cex = trees_ROI$h/20, col = "black")
# corresponding crowns
plot(sf::st_geometry(trees_ROI_crowns), add = TRUE, border = "black", col = NA)
# add ROI
plot(ROI, add = TRUE, border = "red", col = NA)
```

`tree_extraction` *Tree extraction*

Description

creates a data.frame with segment id, height and coordinates of maxima, surface and volume, computed from three images: initial, local maxima and segmented, obtained with `tree_segmentation`. The 2D polygon associated to each crown can be added as a WKT field

Usage

```
tree_extraction(
  r_dem_nl,
  r_maxi = NULL,
  r_dem_w = NULL,
  r_mask = NULL,
  crown = FALSE
)
```

Arguments

`r_dem_nl`	SpatRaster object. Output raster of `tree_segmentation`. Otherwise a raster of canopy height model, preferably filtered to avoid effect of holes on volume and surface computation can be provided. In this case arguments ‘r_maxi‘, ‘r_dem_w‘ have to be provided

`r_maxi`	SpatRaster object. raster with positive values at local maxima (in case ‘r_dem_nl‘ does not contain it)
`r_dem_w`	SpatRaster object. segmented raster (in case ‘r_dem_nl‘ does not contain it)
`r_mask`	SpatRaster object. only segments which maxima are inside the mask are extracted. Values should be NA outside the mask, 1 inside.
`crown`	boolean. Should the 2D crown geometry be added in wkt format to the output data.frame ?

Value

A `sf` collection of POINTs with 7 fields: tree id, local maximum stats (height, dominance radius), segment stats (surface and volume), coordinates (x and y). In case argument ‘crown‘ is ‘TRUE‘, a ‘crown‘ field containing the WKT geometry of the 2D crown is also present. Coordinates are written with one decimal to the right of the order of magnitude of the SpatRaster resolution (e.g. if resolution is 1/3 then 2 decimals are written).

See Also

`tree_segmentation`, `tree_detection`

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# tree segmentation
segments <- tree_segmentation(chm_chablais3)

# tree extraction
trees <- tree_extraction(segments, crown = TRUE)
# create crown polygons from WKT field
trees_crowns <- sf::st_as_sf(sf::st_drop_geometry(trees), wkt = "crown")

# summary of trees without wkt field
summary(trees[, -which(names(trees) == "crown")])

# plot initial image
terra::plot(chm_chablais3, main = "CHM and extracted trees")

# add treetop positions
plot(trees["h"], add = TRUE, cex = trees$h/20, col = "black")
# add crowns
plot(sf::st_geometry(trees_crowns), add = TRUE, border = "black", col = NA)

# plot segments
terra::plot(segments$segments_id, main = "Segments")
# add crowns
plot(sf::st_geometry(trees_crowns), add = TRUE, border = "black", col = NA)
```

`tree_inventory_chablais3`
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(tree_inventory_chablais3)
```

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 abreviated 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

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. pp. 21-22 & 34 https://theses.hal.science/tel-00652698/document

Examples

```
data(tree_inventory_chablais3)
summary(tree_inventory_chablais3)
# display tree inventory
plot_tree_inventory(tree_inventory_chablais3[, c("x", "y")],
  diam = tree_inventory_chablais3$d, col = "red",
  pch = tree_inventory_chablais3$e,
  xlab = "X", ylab = "Y"
)
```

tree_matching — *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

```
tree_matching(lr, ld, delta_ground = 2.1, h_prec = 0.14, stat = 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
`delta_ground`	numeric. buffer around trunk position : absolute value
`h_prec`	numeric. buffer around apex position : proportion of reference tree height
`stat`	boolean. should matching stats be computed

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

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. pp. 53-55 `https://theses.hal.science/tel-00652698/document`

Monnet, J.-M., Mermin, E., Chanussot, J., Berger, F. 2010. Tree top detection using local maxima filtering: a parameter sensitivity analysis. Silvilaser 2010, the 10th International Conference on LiDAR Applications for Assessing Forest Ecosystems, September 14-17, Freiburg, Germany, 9 p. `https://hal.science/hal-00523245/document`

See Also

`plot_matched`, `hist_detection`

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 <- tree_matching(ref_trees, def_trees)
match2 <- tree_matching(ref_trees, def_trees, delta_ground = 2, h_prec = 0)
match1
match2

# 2D display of matching result
plot_matched(ref_trees, def_trees, match1, xlab = "X", ylab = "Y")
plot_matched(ref_trees, def_trees, match2, xlab = "X", ylab = "Y")
```

`tree_segmentation` *Preprocessing and segmentation of raster image for tree identification*

Description

global function for preprocessing (filtering), maxima detection and selection, segmentation and segmentation adjustment of a raster image.

Usage

```
tree_segmentation(dem, dtm = NULL, crown_prop = NULL, crown_hmin = NULL, ...)
```

Arguments

`dem`	raster object or string indicating location of raster file (typically a canopy height model or a digital surface model; in the latter case the dtm parameter should be provided)
`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 are based on the height to terrain.
`crown_prop`	(deprecated) numeric. (overrides `prop` parameter passed to `seg_adjust`, for backward compatibility)
`crown_hmin`	(deprecated) numeric. (overrides `min.value` parameter passed to `seg_adjust`, for backward compatibility)
`...`	arguments passed to functions `dem_filtering` (e.g. `nl_filter`, `nl_size`, `sigma`), `maxima_detection`, `maxima_selection`, `maxima_selection` (dmin: treetop minimum distance to next higher pixel in meters, `dprop`: number defining the treetop minimum distance as proportion of its height to next higher pixel, `hmin`: minimum treetop height), `seg_adjust` (`prop`: minimum height of tree crown base as proportion of treetop height, `min.value`: minimum crown base height)

Value

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

References

Monnet, J.-M. 2011. Using airborne laser scanning for mountain forests mapping: Support vector regression for stand parameters estimation and unsupervised training for treetop detection. Ph.D. thesis. University of Grenoble, France. Section 6.2 https://theses.hal.science/tel-00652698/document

Monnet, J.-M., Mermin, E., Chanussot, J., Berger, F. 2010. Tree top detection using local maxima filtering: a parameter sensitivity analysis. Silvilaser 2010, the 10th International Conference on LiDAR Applications for Assessing Forest Ecosystems, September 14-17, Freiburg, Germany, 9 p. https://hal.science/hal-00523245/document

See Also

dem_filtering, maxima_detection, maxima_selection, segmentation, seg_adjust, tree_extraction, tree_detection

Examples

```
data(chm_chablais3)
chm_chablais3 <- terra::rast(chm_chablais3)

# tree segmentation
segments <- tree_segmentation(chm_chablais3)
segments2 <- tree_segmentation(chm_chablais3,
  nl_filter = "Median", nl_size = 3,
  sigma = cbind(c(0.2, 0.8), c(0, 15)), dmin = 0, dprop = 0, hmin = 10,
  crown_prop = 0.5, crown_hmin = 5
)

# plot initial image segments
terra::plot(chm_chablais3, main = "Initial image")
terra::plot(segments$smoothed_dem, main = "Filtered image")
terra::plot(segments$local_maxima, main = "Local maxima")
#
# replace segment with id 0 (not a tree) with NA
segments$segments_id[segments$segments_id == 0] <- NA
terra::plot(segments$segments_id %% 8, main = "Segments", col = rainbow(8))
#
# plot segmentation with other parameters
segments2$segments_id[segments2$segments_id == 0] <- NA
terra::plot(segments2$segments_id %% 8, main = "Segments2", col = rainbow(8))
```

Index