Package ‘tReeTraits’

February 25, 2026

Type Package

Title Calculate Tree Traits from Terrestrial Lidar

Version 0.1.2

Description Measuring tree architecture from terrestrial lidar data, including tree-level properties, crown characteristics, and structural attributes derived from quantitative structure models (QSMs).

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports data.table, lidR, dplyr, ggplot2, ggpubr, recexcavAAR, rlang, sf, spanner, terra, tibble, reticulate, CrownScorchTLS, FNN, alphashape3d, ggplotify, rgl, stats, stringr, readr

NeedsCompilation no

Author Jeffery B. Cannon [aut, cre]

Maintainer Jeffery B. Cannon <`jeffery.cannon@jonesctr.org`>

Repository CRAN

Date/Publication 2026-02-25 19:00:02 UTC

Contents

`.read_qsm_raw` *Read a PyTLidar QSM file*

Description

Reads a PyTLidar-generated cylinder file and converts it into a tidy data frame with start/end coordinates, radius, length, volume, and branching order.

Usage

```
.read_qsm_raw(cyl_file)
```

Arguments

`cyl_file`	Path to the PyTLidar cylinder output file (.txt).

Value

A data frame with columns startX, startY, startZ, endX, endY, endZ, cyl_ID, parent_ID, extension_ID, radius_cyl, length, volume, branching_order.

`add_verticality` *Add point-wise verticality from local PCA*

Description

Computes a verticality metric (0-1) for each point in a LAS object based on the z-component of the dominant local PCA eigenvector.

Usage

```
add_verticality(las, k = 30, name = "verticality")
```

Arguments

`las`	A LAS object.
`k`	Number of nearest neighbors for local PCA.
`name`	Name of the attribute to store.

Value

The input ‘LAS‘ object with a new numeric attribute containing verticality values (0–1).

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = add_verticality(las, k = 20)
head(las@data)
```

`basics_diagnostic_plot` *Diagnostic Plot of Basic Tree Measurements*

Description

Generates a diagnostic 2D plot showing basic tree measurements such as tree height, crown base height (CBH), crown width, and diameter at breast height (DBH) over a subsampled LAS point cloud.

Usage

```
basics_diagnostic_plot(las, height, cbh, crown_width, dbh, res = 0.1)
```

Arguments

`las`	A ‘LAS‘ object (from the ‘lidR‘ package) containing the tree point cloud.
`height`	Numeric. Total tree height.
`cbh`	Numeric. Crown base height.
`crown_width`	Numeric. Crown width.
`dbh`	Numeric. Diameter at breast height.
`res`	Numeric. Resolution for point cloud thinning; default is 0.1 m.

Details

The function first thins the point cloud using ‘lidR::decimate_points()‘ to improve plotting speed. Then it overlays key measurement lines and markers for height, crown width, DBH, and crown base height.

Value

A ‘ggplot‘ object displaying thinned tree points and markers for key measurements.

Examples

```
library(lidR)
path = system.file('extdata', 'tree_0744.laz', package='tReeTraits')
las = clean_las(readLAS(path))
ht = get_height(las)
dbh = get_dbh(las)
wid = get_width(las)[1]
cbh = get_crown_base(las)
basics_diagnostic_plot(las, height = ht, cbh = cbh,
                     crown_width = wid, dbh = dbh)
```

`branch_distribution_plot`

Diagnostic Plot of Branch Diameter Distribution

Description

Generates a diagnostic plot showing the distribution of branch diameters in a QSM. This is a wrapper around ‘branch_size_distribution()‘ which computes branch metrics.

Usage

```
branch_distribution_plot(qsm)
```

Arguments

`qsm`	A QSM object (e.g., data frame returned by ‘run_treeqsm()‘) containing cylinder information.

Details

The function calls ‘branch_size_distribution()‘ with ‘plot = FALSE‘ to compute branch volumes at midpoints of diameter bins, then generates a bar plot showing total volume per diameter bin.

Value

A ‘ggplot‘ object displaying branch diameter (x-axis) versus total branch volume (y-axis).

Examples

```
qsm_file = system.file('extdata',"tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
branch_distribution_plot(qsm)
```

`branch_size_distribution` *Get Branch size distribution from a QSM*

Description

This function outputs the volume (in mL) distribution of branches across different branch diameter classes (in cm). Outputs a table that can be used in other functions to find branch_skewness or branch_volume_weighted_stats()

Usage

```
branch_size_distribution(qsm, breaks = NULL, plot = TRUE)
```

Arguments

`qsm`	– a QSM loaded using ‘[load_qsm()]‘.
`breaks`	numeric – a vector of diameter classes (in cm) by which to summarize branch volume. If ‘NULL‘ the branch of branch sizes will be distributed across 1 cm bins.
`plot`	boolean – indicates whether the branch diameter distribution should be plotted as a histogram.

Value

A tibble summarizing branch volume by diameter class with columns:

diameter_cm Factor indicating the diameter class (cm).

midpoint Numeric midpoint (cm) of each diameter class.

volume_mL Total branch volume (mL) within the class.

Returns ‘NA‘ if fewer than two branch cylinders are present.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
branch_distribution = branch_size_distribution(qsm, plot=TRUE)
print(branch_distribution)

#volume-weighted mean
branch_volume_weighted_stats(qsm, FUN = function(x) mean(x))

#volume-weighted median
branch_volume_weighted_stats(qsm, FUN = function(x) median(x))

# volume-weighted skewness
branch_volume_weighted_stats(qsm, FUN = function(x) 3*(mean(x) - median(x)) / sd(x))
```

`branch_volume_weighted_stats`
Calculate volume weighted branch statistics

Description

This function calculates statistics on branch diameters weighted by the volume of branches of that size based on outputs from ‘branch_size_distribution()‘. The user defined function ‘FUN‘ can take any form of f(x) where x is a vector of diameters of length 1 for every mL of volume of that branch size class. See Details for recommended values for ‘FUN‘. #Details Values of central tendency are recommended, but not variance since the weighted means are simulated.

Usage

```
branch_volume_weighted_stats(qsm, breaks = NULL, FUN = function(x) mean(x))
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.
`breaks`	numeric – a vector of diameter classes (in cm) by which to summarize branch volume. If ‘NULL‘ the branch of branch sizes will be distributed across 1 cm bins.
`FUN`	function – central tendency function to be weighted based on branch volume.

Details

Recommended values of ‘FUN‘ are:

Mean FUN = function(x) mean(x)

Median FUN = function(x) median(x)

Skewness FUN = function(x) 3*(mean(x) - median(x)) / sd(x)

Value

A numeric value representing the volume-weighted statistic calculated by ‘FUN‘ across branch diameter classes.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
branch_distribution = branch_size_distribution(qsm, plot=TRUE)
print(branch_distribution)

#volume-weighted mean
branch_volume_weighted_stats(qsm, FUN = function(x) mean(x))

#volume-weighted median
branch_volume_weighted_stats(qsm, FUN = function(x) median(x))

# volume-weighted skewness
branch_volume_weighted_stats(qsm, FUN = function(x) 3*(mean(x) - median(x)) / sd(x))
```

`clean_las` *Load, Recenter, and Remove low vegetation from ‘LAS‘ object representing segmented tree*

Description

Function to normalize, remove noise, remove vegetation, and recenter ‘LAS‘ representing segmented tree. Vegetation cleaning is accomplished by identifying stem points (CrownScrochTLS::StemPoints) and removing all but the Stem below the ‘z.threshold‘.

Usage

```
clean_las(las, bole_height = 1, quantile = 0.001)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`bole_height`	numeric, height threshold below which all stem points can be considered vegetation.
`quantile`	See ‘normalize_las‘. Z quantile at which grown level is specified since ground points may not be identifiable with common algorithms if ground points are removed during segmentation#’

Value

A cleaned ‘LAS‘ object with vegetation and noise removed, normalized and recentered.

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las_cleaned = clean_las(las)

plot(las)
plot(las_cleaned)
```

---

convex_hull_2D *Returns the convex hull representing vertical crown area*

Description

This function generates an ‘sf‘ object representing th vertical crown area of a ‘LAS‘ object based using the convex hull of a 2D vertical projection.

Usage

```
convex_hull_2D(las, angle = 0)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing the CROWN of a tree. Crowns must be segmented using [segment_crown()].
`angle`	numeric - in degrees, rotation angle about Z axis.

Value

An ‘sf‘ polygon representing the vertical convex hull of the crown projection.

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
get_crown_volume_voxel(las)
get_crown_volume_alpha(las)
sf::st_area(convex_hull_2D(las)) #profile area, convex hull
sf::st_area(voxel_hull_2D(las)) #profile area, voxel hull
get_lacunarity(las)
```

`fit_taper_Kozak` *Fit taper equation to QSM*

Description

This function fits a taper equation to trunk sections of a QSM using Kozak the Model (2002, 2007).

Usage

```
fit_taper_Kozak(
  qsm,
  dbh,
  terminus_diam_cm = 4,
  segment_size = 0.25,
  plot = TRUE
)
```

Arguments

`qsm`	a QSM loaded using '[load_qsm()]'.
`dbh`	numeric – required to fit Kozak model, not calculated from QSM, so as not to conflict with other more accurate means of measurement e.g., 'get_DBH'
`terminus_diam_cm`	numeric – the trunk diameter at which is no longer considered trunk
`segment_size`	numeric – the length of segments that QSM cylinders are grouped into
`plot`	boolean – indicates whether model output should be plotted. Plots are found in the output list as object$plot, regardless of this setting.

Details

$ d(h)/D = a0 (h/H) + a1 (h/H) + a2 (h/H)^2 + a3 (h/H)^3 $

The function groups QSM cylinders into segments of 'segment_size' up to 'terminus_diam' which is the maximum diameter at which the taper equation ends.

Value

A list with components:

data Tibble of trunk segment heights and observed diameters used in model fitting.

plot A 'ggplot2' object showing observed diameters and fitted taper curve.

results Data frame containing fitted Kozak parameters ('a0'–'a3'), coefficient of determination ('r2'), and root mean squared error ('rmse').

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
fit_taper_Kozak(qsm, dbh = 13.8)
```

`full_diagnostic_plot` *Generate a diagnostic plot to assess basic metrics and QSM output*

Description

Generate a diagnostic plot to assess basic metrics and QSM output

Usage

```
full_diagnostic_plot(las, qsm, height, cbh, crown_width, dbh, res = 0.1)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing the CROWN of a tree. Crowns can be segmented using [segment_crown()]
`qsm`	a QSM loaded using ‘[load_qsm()]‘.
`height`	numeric - tree height, or generated from ‘get_height()‘
`cbh`	numeric - crown base heigth, or generated from ‘get_crown_base()‘.
`crown_width`	numeric - crown width height, or generated from ‘get_width()‘
`dbh`	numeric - in cm, tree diameter at breast height, or generated from ‘get_dbh()‘
`res`	numeric - resolution of voxelization to speed up plotting

Value

A multi-panel ‘ggplot‘ object (class ‘ggarrange‘) summarizing tree structural metrics and QSM diagnostics, suitable for printing or saving with ‘ggplot2‘.

Examples

```
library(lidR)
las_file = system.file("extdata", "tree_0744.laz", package="tReeTraits")
las = lidR::readLAS(las_file, filter = '-thin_with_voxel 0.1')
las = clean_las(las, bole_height=3)
height = get_height(las)
crown_width = get_width(las)
dbh = get_dbh(las, select_n=30)
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
full_diagnostic_plot(las=las, qsm=qsm, height=height, cbh=cbh, crown_width=crown_width, dbh=dbh)
```

`get_area_profile` *Generate area estimates of tree profile in segments*

Description

This function calculates the area of the tree profile by breaking it into segments of height ‘segment_height‘ and estimating the width of each segement. Area profiles are useful for caluclating total area, but also used to detect crown base height.

Usage

```
get_area_profile(las, segment_height = 0.25, quantile = c(0.001), angle = 0)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree, with the crown labeled.
`segment_height`	numeric - height of each segment in which to calculate area
`quantile`	numeric - quantile at which width is measured Values in the interval approaching 0 (e.g., 0.001) are recommended to trim random noise
`angle`	numeric - angle at which to rotate the point cloud prior to estimating area. Useful in a loop if quantifying mulitple angles

Value

A tibble with columns ‘bottom‘, ‘top‘, ‘width‘, ‘area‘, and ‘angle‘ describing the vertical area profile.

`get_center_of_mass` *Get center of mass from tree bole segments QSM (X, Y, Z)*

Description

This function calculates the center of mass/volume from a QSM by estimating the centroid of cylinder locations, each weighted by their volume. Only trunk sections are included (e.g., ‘branching_order == 0‘). For center of mass, assumes constant density within segments.

Usage

```
get_center_of_mass(qsm)
```

Arguments

`qsm`	qsm object loaded from ‘[load_qsm]‘.

Value

A tibble with one row and three numeric columns:

X Volume-weighted X coordinate of the trunk center of mass (m).

Y Volume-weighted Y coordinate of the trunk center of mass (m).

Z Volume-weighted Z coordinate of the trunk center of mass (m).

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
print(get_center_of_mass(qsm))
```

`get_com_offset` *Horizontal offset of center of mass from QSM*

Description

This function extracts the horizontal distance between the base of a tree and the center of a tree from a QSM. The function takes the coordinate of the lowest QSM segment, and the center of mass, and finds the horizontal distance between them.

Usage

```
get_com_offset(qsm)
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.

Value

A single numeric value giving the horizontal distance (m) between the base of the tree and the volume-weighted center of mass.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
print(get_center_of_mass(qsm))
print(get_com_offset(qsm))
```

`get_crown_base` *Estimate Crown base height of ‘LAS‘ representing segmented tree.*

Description

This function estimates the crown base height by analyzing the vertical profile of the tree using [get_area_profile()] which breaks the profile into segments of height ‘segment_height‘. The function estimates segments exceed a threshold specified by ‘threshold‘ which must be exceeded ‘sustain‘ times.

Usage

```
get_crown_base(
  las,
  threshold = 0.5,
  sustain = 2,
  segment_height = 0.25,
  quantile = 0.01
)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`threshold`	numeric - threshold width at which crown becomes apparent. Recommend a width ~2X greater than anticipated DBH.
`sustain`	numeric - number of segments in a row that treshold must be exceeded before identifying start of crown. This is to exclude small segments of crown isolated from larger continuous crown.
`segment_height`	numeric - height of each segment in which to calculate area
`quantile`	numeric - quantile at which width is measured Values in the interval approaching 0 (e.g., 0.001) are recommended to trim random noise

Value

A named numeric vector with element ‘crown_base_height‘ (m).

Examples

```
# example code
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package = "tReeTraits"))
las = clean_las(las)

# Estimate crown base height
cbh = get_crown_base(las)
print(cbh)
```

`get_crown_lever_arm` *Calculate crown leverage from point cloud*

Description

This function calculates the lever arm of canopies. The function #' is a simple wrapper for 'get_area_profile()'. It calculates the crown area in segments defined by 'segement_height', multiplies the area of each of those segments by their height, and then returns the sum of all segments. This is proportaionl to drag calculations on the tree assuming windspeed is invariant with height.

Usage

```
get_crown_lever_arm(las, segment_height = 0.25, quantile = c(0.001), angle = 0)
```

Arguments

`las`	'LAS' object from 'lidR' package representing individually segmented tree, with the crown labeled. See 'segment_crown()'
`segment_height`	numeric - height of each segment in which to calculate area
`quantile`	numeric - quantile at which width is measured Values in the interval approaching 0 (e.g., 0.001) are recommended to trim random noise
`angle`	numeric - angle at which to rotate the point cloud prior to estimating area. Useful in a loop if quantifying mulitple angles

Value

A numeric value representing the crown lever arm.

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
las = segment_crown(las)
print(get_crown_lever_arm(las))
```

`get_crown_volume_alpha` *Estimate crown volume by alpha shape volume*

Description

This function volume of a 'LAS' object by thinning to a resolution specified by 'resolution', and estimating volume by fitting a alpha shape volume. Crowns must be segmented using [segment_crown()].

Usage

```
get_crown_volume_alpha(las, resolution = 0.1, alpha = 0.5)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing a tree. Crowns must be segmented using [segment_crown()].
`resolution`	numeric - resolution of initial voxelization to increase speed
`alpha`	numeric - alpha for the computation of the 3D alpha-shape of the point cloud. See [alphashape3d::ashape3d].

Value

A named numeric vector with element ‘crown_volume_alpha‘ (m^3).

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
get_crown_volume_voxel(las)
get_crown_volume_alpha(las)
sf::st_area(convex_hull_2D(las)) #profile area, convex hull
sf::st_area(voxel_hull_2D(las)) #profile area, voxel hull
get_lacunarity(las)
```

`get_crown_volume_voxel`
Estimate crown volume by voxelization

Description

This function volume of a ‘LAS‘ object by thinning to a resolution specified by ‘resolution‘, and estimating volume using the equation

$$Volume_{crown} = N_{occupiedvoxel} * Volume_{voxel}$$

Usage

```
get_crown_volume_voxel(las, resolution = 0.1)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing a tree. Crowns must be segmented using [segment_crown()].
`resolution`	numeric - resolution of voxelization

Value

A named numeric vector with element ‘crown_volume_vox‘ (m^3).

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
get_crown_volume_voxel(las)
get_crown_volume_alpha(las)
sf::st_area(convex_hull_2D(las)) #profile area, convex hull
sf::st_area(voxel_hull_2D(las)) #profile area, voxel hull
get_lacunarity(las)
```

---

get_dbh — *Extract diameter at breast height from ‘LAS‘ object representing segmented tree.*

Description

Function to extract diameter at breast height (1.37 m) from LAS object. Function filters LAS keeping only points with Intensity greater than specified threshold. Function calculates verticality eigenvalue and filters based on verticality threshold. Last, diameter is calculated using a RANSAC cylinder fitting algorithm.

Usage

```
get_dbh(
  las,
  intensity_threshold = 41000,
  select_n = 10,
  verticality_threshold = 0.9
)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`intensity_threshold`	numeric - filter value for Intensity to help remove vegetation
`select_n`	numeric - number of points selected on every RANSAC iteration.
`verticality_threshold`	numeric - filter value for Verticality threshold to remove horizontal branches.

Value

A named numeric vector with element ‘dbh‘ (m).

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
print(get_dbh(las))
```

get_height *Extract height from ‘LAS‘ object representing segmented tree.*

Description

Function to extract height from LAS object. Function calculates difference between two specified quantiles from the ‘Z‘ attribute.

Usage

```
get_height(las, quantiles = c(0, 1))
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`quantiles`	Z quantiles at which ground level and highest point are measured. Values in the interval (0,1) are recommended to trim random noise.

Value

A named numeric vector with element ‘height‘ (m).

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
print(get_height(las))
```

get_lacunarity *Calculate crown lacunarity from a tree crown*

Description

This function calculates the lacunarity or "porosity" of a tree crown by defined as 1 - the ratio of a voxelized crown hull and a convex hull. See ‘voxel_hull_2D()‘ and ‘convex_hull_2D()‘

Usage

```
get_lacunarity(las, res = 0.1, angle = 0)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing the CROWN of a tree. Crowns must be segmented using [segment_crown()].
`res`	numeric - resolution of voxelization
`angle`	numeric - in degrees, rotation angle about Z axis.

Value

A numeric value representing crown lacunarity (unitless).

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
get_crown_volume_voxel(las)
get_crown_volume_alpha(las)
sf::st_area(convex_hull_2D(las)) #profile area, convex hull
sf::st_area(voxel_hull_2D(las)) #profile area, voxel hull
get_lacunarity(las)
```

`get_primary_branches` *Extract primary branches from a QSM*

Description

Extract primary branches from a QSM by filtering out the the trunk, and identifying all cylinders where ‘branching_order == 1‘ and are attached to the trunk. Returns a tibble containing the basal diameter and height of attachment points.

Usage

```
get_primary_branches(qsm)
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.

Value

A tibble with one row per primary branch containing:

section Character string ("branches").

diam_cm Basal diameter of the branch (cm).

ht_m Height of branch attachment (m).

volume Placeholder column (currently ‘NA‘).

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
primary_branches = get_primary_branches(qsm)
#number of primary branches
nrow(primary_branches)
```

`get_stem_sweep` *Calculate tree sweep from straight line from QMS*

Description

This function calculates tree sweep from a QSM. Starting with an idealized vector of a straight tree (straight line from top to bottom QSM segment) the function caclucates devations of points along the trunk from the idealized vector. It resturns sweep from each QSM segment so that summary statistics can be computed by the user. See also ‘get_stem_deflection()‘

Usage

```
get_stem_sweep(qsm, terminus_diam_cm = 4, plot = TRUE)
```

Arguments

`qsm`	QSM loaded using ‘[load_qsm()]‘.
`terminus_diam_cm`	numeric – the trunk diameter at which is no longer considered trunk
`plot`	boolean – indicates whether graph of sweep should be plotted.

Value

A data frame with columns:

Height Height (m) of each trunk segment midpoint.

sweep Perpendicular deviation (m) from the idealized straight stem line.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
print(get_center_of_mass(qsm))
print(get_com_offset(qsm))
```

get_stem_tilt *Get tree tilt from QSM*

Description

This function calculates tilt of a tree from a QSM. The function identifies the upper and lower extreme segments of the QMS (trunk sections only) and computes a vector between them, and returns the devation of that angle from directly vertical.

Usage

```
get_stem_tilt(qsm, terminus_diam_cm = 4)
```

Arguments

`qsm`	a QSM loaded using '[load_qsm()]'.
`terminus_diam_cm`	numeric - trunk diameter at which it is treated as a branch.

Value

A single numeric value giving stem tilt in degrees from vertical.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
get_stem_tilt(qsm)
```

get_width *Extract width from 'LAS' object representing segmented tree.*

Description

Function to extract width from LAS object. Function calculates difference between two specified quantiles from the 'X' and 'Y' attributes and returns both widths and their average.

Usage

```
get_width(las, quantiles = c(0.001, 0.999))
```

Arguments

`las`	'LAS' object from 'lidR' package representing individually segmented tree
`quantiles`	Z quantiles at which widths are measured are measured. Values in the interval (0,1) are recommended to trim random noise.

Value

A named numeric vector with elements ‘mean_width‘, ‘x_width‘, and ‘y_width‘ (m).

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
print(get_width(las))
```

`hull_diagnostic_plot` *Diagnostic Plot of Crown Convex Hulls*

Description

Generates a 2D diagnostic plot to visualize the convex hulls and voxelized hulls of tree crowns in a LAS point cloud. Useful for checking results of crown segmentation.

Usage

```
hull_diagnostic_plot(las, res = 0.1)
```

Arguments

`las`	A ‘LAS‘ object from the ‘lidR‘ package. Must contain a column named ‘Crown‘.
`res`	Numeric. Resolution for voxelization in ‘voxel_hull_2D()‘. Default is 0.1.

Details

The function first filters points marked as crown (‘Crown == 1‘) and then computes both the 2D convex hull and a voxelized 2D hull. The resulting plot overlays the voxel hull in color and the convex hull as a dashed outline.

Value

A ‘ggplot‘ object displaying convex hulls (dashed) and voxel hulls (filled).

Examples

```
library(lidR)
file = system.file('extdata', file='tree_0744.laz', package='tReeTraits')
las <- readLAS(file)
las <- segment_crown(las)  # adds `Crown` column
hull_diagnostic_plot(las)
```

`internode_distances` *Get internode distances between primary branches from a QSM*

Description

This function estimates the internode distance between primary branches from a QSM. It filters out all primary branches ‘branching_order == 1‘ calculates their attachment points (Z) to the trunk, and then returns the distances betweent the branches

Usage

```
internode_distances(qsm, min_diam = 2)
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.
`min_diam`	numeric - minimum diameter (in cm) to include branch

Value

A numeric vector of internode distances (m) between consecutive primary branches that meet the diameter threshold. Returns ‘NA‘ if fewer than two qualifying primary branches are present.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
inodes = internode_distances(qsm)
print(inodes)
median(inodes)
```

`load_qsm` *Load and Validate a QSM File*

Description

Reads a Quantitative Structure Model (QSM) file from disk and checks that it includes all required columns. If any expected columns are missing, the function stops with an informative error message.

Usage

```
load_qsm(path)
```

Arguments

`path`	Character string giving the path to a QSM text file.

Value

A tibble containing the QSM data.

A tibble containing the QSM data with validated required columns: ‘startX‘, ‘startY‘, ‘startZ‘, ‘endX‘, ‘endY‘, ‘endZ‘, ‘cyl_ID‘, ‘parent_ID‘, ‘extension_ID‘, ‘radius_cyl‘, ‘length‘, ‘volume‘, and ‘branching_order‘. An error is thrown if any required columns are missing.

Examples

```
qsm_path = system.file('extdata', 'tree_0744_qsm.txt', package='tReeTraits')
qsm <- load_qsm(qsm_path)
plot_qsm2d(qsm, scale=50)
```

normalize_las *Normalize ‘LAS‘ object representing segmented tree.*

Description

Function to normalize LAS object. Function calculates ground level based on the parameter specified by ‘quantile‘, subtracts it from all ‘Z‘.

Usage

```
normalize_las(las, quantile = c(0.001))
```

Arguments

las	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
quantile	Z quantile at which grown level is specified since ground points may not be identifiable with common algorithms if ground points are removed during segmentation

Value

A ‘LAS‘ object with Z values normalized to ground level.

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
# view histogram of Z values ranging from  -18 to -7 m
hist(las$Z)
las = normalize_las(las)
# view histogram of Z values now ranging from  0 to 11 m
hist(las$Z)
```

`plot_qsm2d` *Plot QSM in base R*

Description

Simple function to create a diagnostic plot to view QSMs colored by branching order.

Usage

```
plot_qsm2d(qsm, scale = 150, rotation = TRUE)
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.
`scale`	a factor by which to multiply the ‘radius_cyl‘ column to give line segments the appearance of volume
`rotation`	boolean - indicates whether the plot should display the tree from 2 angles TRUE, or just one FALSE.

Value

‘NULL‘, invisibly. Produces a base R plot as a side effect.

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
plot_qsm2d(qsm)
```

`plot_qsm3d` *Plot QSM Cylinders in 3D*

Description

Renders a 3D visualization of tree cylinders (from a Quantitative Structure Model, QSM) using actual geometric radii. Each cylinder is drawn between its start and end coordinates with the radius provided in the QSM data.

Usage

```
plot_qsm3d(qsm, bg = "white", color = "black", alpha = 0.7)
```

Arguments

`qsm`	A data frame or tibble containing QSM cylinder data with columns: `startX`, `startY`, `startZ`, `endX`, `endY`, `endZ`, and `radius_cyl`.
`bg`	Background color for the 3D plot. Defaults to `"white"`.
`color`	Cylinder color. Defaults to `"black"`.
`alpha`	Transparency level for cylinders, between 0 (fully transparent) and 1 (fully opaque). Defaults to `0.7`.

Details

This function uses **rgl** to draw 3D cylinders representing each segment of a tree model. It is intended for visualizing QSM output such as that produced by **PyTLidar** or other tree reconstruction algorithms.

For large models, rendering may be slow because each cylinder is drawn as a separate mesh. Consider downsampling or filtering before plotting.

Value

Opens an interactive 3D rgl window with rendered cylinders. Returns `NULL` invisibly.

A ‘ggplot‘ object combining multiple diagnostic panels.

Examples

```
# Load QSM output (example path)
qsm_file = system.file('extdata',"tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
# Plot with real radii

plot_qsm3d(qsm, color = "forestgreen", alpha = 0.6)
```

`plot_tree` *Make 3-panel plot of tree point cloud to check for errors*

Description

Plots 2 profiles X, Y, and and overhead Z view of a point cloud to allow users to identify stray points, or errors in segmentations.

Usage

```
plot_tree(las, res = 0.05, plot = TRUE)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing the CROWN of a tree. Crowns can be segmented using [segment_crown()].
`res`	numeric - resolution of voxelization to speed up plotting
`plot`	boolean - indicates whether to print the output plot, in both cases a ggplot object is returned in the output.

Value

A ‘ggplot‘ object containing the arranged diagnostic panels.

Examples

```
# example code
library(lidR)
file = system.file("extdata", "tree_0744.laz", package="tReeTraits")
las = readLAS(file, filter = '-thin_with_voxel 0.1')
las = clean_las(las)
plot_tree(las)
```

---

qsm_volume_distribution

Volume distribution from QSM

Description

This function estimates tree volume and its vertical distribution from a QSM. The function separates the QSM into (1) trunk sections (2) terminus (top of trunk < 4 cm dbh), and (3) primary branches. The function divides trunk into segments defined by ‘segment_size‘, calculates QSM volume, For tree portions identified as branches the function only returns the diameter. Both of these can be used in mass-volume equations as needed.

Usage

```
qsm_volume_distribution(qsm, terminus_diam_cm = 4, segment_size = 0.5)
```

Arguments

`qsm`	a QSM loaded using ‘[load_qsm()]‘.
`terminus_diam_cm`	numeric - trunk diameter at which it is treated as a branch.
`segment_size`	numeric length of trunk segments in which to summarize volume.

Value

A tibble describing vertical volume distribution with columns:

section Tree component ("trunk", "terminus", or "branches").

diam_cm Diameter (cm) of the segment or branch.

ht_m Height (m) of the segment midpoint or branch attachment.

volume Total volume (m^3) of the segment (trunk and terminus only; ‘NA‘ for branches).

Examples

```
qsm_file = system.file("extdata", "tree_0744_qsm.txt", package='tReeTraits')
qsm = load_qsm(qsm_file)
volume = qsm_volume_distribution(qsm)
print(volume)
plot(volume~ht_m, data=volume, type='l', xlab='height (m)', ylab='Volume (m3)')
```

`recenter_las` *Recenter ‘LAS‘ object representing segmented tree based on the bole location*

Description

Function calculates the tree location using points below specified ‘height‘ and recenters on ‘X=0 Y=0‘

Usage

```
recenter_las(las, height = 1)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing
`height`	consider only points where Z < height, if specified. Useful for considering only the tree bole, for centering. individually segmented tree. Set ‘height = NULL‘ to recenter using all points.

Value

A ‘LAS‘ object with X and Y coordinates recentered to (0, 0).

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
# view histogram of original X/Y values
par(mfrow=c(1,2))
hist(las$X)
hist(las$Y)
las = recenter_las(las)
# view histogram of X/Y values centered on 0,0
hist(las$X)

hist(las$Y)
```

rotate_las_z *Rotate ‘LAS‘ object about the ‘Z‘ axis*

Description

Rotate ‘LAS‘ object about the ‘Z‘ axis for specified angle.

Usage

```
rotate_las_z(las, angle)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`angle`	numeric - in degrees, rotation angle about Z axis.

Value

A ‘LAS‘ object rotated about the Z axis.

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las_rotated = rotate_las_z(las, 90)

plot(las)
plot(las_rotated)
```

run_treeqsm *Run TreeQSM on a LAS/LAZ file using PyTLidar*

Description

Processes a point cloud, filters and normalizes it, then runs the PyTLidar TreeQSM model to reconstruct cylinders representing the tree structure.

Usage

```
run_treeqsm(
  file,
  output_dir = NULL,
  intensity_threshold = 40000,
  resolution = 0.02,
  patch_diam1 = c(0.05, 0.1),
  patch_diam2min = c(0.04, 0.05),
  patch_diam2max = c(0.12, 0.14),
  optimizing_metrics = c("TrunkMean", "Branch1Mean"),
  verbose = TRUE
)
```

Arguments

`file`	Path to the input LAS or LAZ file.
`output_dir`	Directory to save output files; temporary if NULL.
`intensity_threshold`	Minimum point intensity to retain.
`resolution`	Thinning voxel size (m).
`patch_diam1`	Numeric vector of patch diameter 1 parameters.
`patch_diam2min`	Numeric vector of minimum patch diameter 2.
`patch_diam2max`	Numeric vector of maximum patch diameter 2.
`optimizing_metrics`	Character vector of metric names to average and minimize when selecting the best QSM fit. See Details
`verbose`	Logical; whether to print details during processing.

Details

The `optimizing_metrics` argument controls which point-to-cylinder distance summaries are used to evaluate TreeQSM fits. These metrics quantify how closely reconstructed cylinders match the underlying point cloud and are computed for different structural components of the tree.

Available metrics include:

- `median`, `mean`, `max`, `std`: Overall point–to–cylinder distances.

- `TrunkMedian`, `TrunkMean`, `TrunkMax`, `TrunkStd`: Trunk-only distances.
- `BranchMedian`, `BranchMean`, `BranchMax`, `BranchStd`: All branch distances.
- `Branch1Median`, `Branch1Mean`, `Branch1Max`, `Branch1Std`: First-order branch distances.
- `Branch2Median`, `Branch2Mean`, `Branch2Max`, `Branch2Std`: Second-order branch distances.

When multiple metrics are supplied, their row-wise mean is computed and minimized to select the best-fitting QSM, allowing users to balance fit quality across different tree components.

Value

A list with elements:

- `qsm_pars`: Data frame of selected patch parameters and fit metrics.
- `qsm`: Data frame of cylinder-level QSM output.

A list with elements:

qsm_pars Data frame of patch parameters and fit metrics.

qsm Data frame of cylinder-level QSM output.

Examples

```
## Not run:
file <- system.file("extdata", "tree_0744.laz", package="tReeTraits")
run_treeqsm(file)

## End(Not run)
```

segment_crown *Segment tree crown of ‘LAS‘ representing segmented tree.*

Description

This function labels all points with $Z > ‘crown_base_height‘$ and returns a labled LAS. If ‘crown_base_height‘ is not specified, it is estimated with [get_crown_base()] using default parameters.

Usage

```
segment_crown(las, crown_base_height = NULL)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing individually segmented tree
`crown_base_height`	numeric - height of crown base for segmentation. ‘NULL‘, it is estimated with [get_crown_base()] using default parameters.

Value

The input ‘LAS‘ object with a new attribute ‘Crown‘ (1 = crown, 0 = non-crown).

Examples

```
library(lidR)
las = readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
las = segment_crown(las)

#Plot with color based on crown
plot(las, color='Crown')
```

`setup_pytlidar` *Setup PyTLidar Python environment*

Description

Ensures Python 3.11 is available via conda, creates ’r-reticulate-pytlidar’ environment if needed, activates it, and installs required Python modules for PyTLidar.

Usage

```
setup_pytlidar()
```

Details

This function **must be called manually** before using any Python-dependent functions.

Value

‘TRUE‘ if the environment is successfully activated and all required Python modules are available, ‘FALSE‘ if installation of Miniconda was required and R should be restarted.

Examples

```
## Not run:
setup_pytlidar()

## End(Not run)
```

`taper_diagnostic_plot` *Diagnostic Plot of Tree Taper*

Description

Generates a diagnostic plot showing the fitted taper of a tree using the Kozak model. This is a simple wrapper around ‘fit_taper_Kozak()‘ to extract its plot output.

Usage

```
taper_diagnostic_plot(qsm, dbh)
```

Arguments

`qsm`	A QSM object (e.g., data frame returned by ‘run_treeqsm()‘) containing cylinder information.
`dbh`	Numeric. Diameter at breast height of the tree, used as input to the taper function.

Details

The function calls ‘fit_taper_Kozak()‘ with ‘plot = FALSE‘ and returns the plot component. This allows quick visualization of the taper without modifying the underlying QSM.

Value

A ‘ggplot‘ object showing the fitted taper along the tree stem.

Examples

```
path = system.file('extdata', 'tree_0744_qsm.txt', package='tReeTraits')
qsm = load_qsm(path)
taper_diagnostic_plot(qsm, dbh = 0.25)
```

`voxel_hull_2D` *Returns an ‘sf‘representing the vertical crown area from voxelization*

Description

This function generates an ‘sf‘ object representing th vertical crown area of a ‘LAS‘ object by voxelizing a 2D vertical projection.

Usage

```
voxel_hull_2D(las, resolution = 0.1, angle = 0)
```

Arguments

`las`	‘LAS‘ object from ‘lidR‘ package representing the CROWN of a tree. Crowns must be segmented using [segment_crown()].
`resolution`	numeric - resolution of voxelization
`angle`	numeric - in degrees, rotation angle about Z axis.

Value

An ‘sf‘ polygon representing the voxel-based vertical crown hull.

Examples

```
las = lidR::readLAS(system.file("extdata", "tree_0744.laz", package="tReeTraits"))
las = clean_las(las)
cbh = get_crown_base(las, threshold=0.25, sustain=2)
las = segment_crown(las, cbh)
get_crown_volume_voxel(las)
get_crown_volume_alpha(las)
sf::st_area(convex_hull_2D(las)) #profile area, convex hull
sf::st_area(voxel_hull_2D(las)) #profile area, voxel hull
get_lacunarity(las)
```

`write_qsm` *Save QSM results and patch parameters*

Description

Writes QSM cylinder data and parameter summaries to tab-delimited text files.

Usage

```
write_qsm(qsm, name, output_dir = getwd())
```

Arguments

`qsm`	Output from `run_treeqsm()`; a list with `qsm` and `qsm_pars`.
`name`	Base name to use for output files.
`output_dir`	Directory where files are written (defaults to current working directory).

Value

Invisibly returns a list with file paths:

- `qsm`: Path to the QSM cylinder file.
- `pars`: Path to the parameter summary file.

Examples

```
## Not run:
# Run and Load a qsm from a laz file.
# ---- Step 1. Define input file  ----
# Input file
file <- system.file("extdata", "tree_0744.laz", package = "tReeTraits")
tree_id <- tools::file_path_sans_ext(basename(file))

# ---- Step 2. Run TreeQSM ----
# Multiple parameter combinations can be supplied; TreeQSM optimizes across them
qsm_result <- run_treeqsm(
  file = file,
  intensity_threshold = 40000,
  resolution = 0.02,
  patch_diam1 = c(0.05, 0.1),
  patch_diam2min = c(0.04, 0.05),
  patch_diam2max = c(0.12, 0.14),
  verbose = TRUE
)

# ---- Step 3. Save results ----
write_qsm(
  qsm_result,
  name = tree_id,
  output_dir = tempdir()
)

# ---- Step 4. Reload QSM ----
qsm_path <- file.path(tempdir(), paste0(tree_id, "_qsm.txt"))
qsm <- load_qsm(qsm_path)

# ---- Step 5. Visualize ----
plot_qsm2d(qsm, scale = 50)
plot_qsm3d(qsm)
## End(Not run)
```

Index